МІНІСТЕРСТВО ОСВІТИ І НАУКИ УКРАЇНИ
НАЦІОНАЛЬНИЙ ТЕХНІЧНИЙ УНІВЕРСИТЕТ УКРАЇНИ
«Київський політехнічний інститут імені Ігоря Сікорського»

С.В.Кисляк, Є.А.Настенко

Основи молекулярної біології та біоінформатики

комп`ютерний практикум

Рекомендовано Методичною радою КПІ ім. Ігоря Сікорського
як навчальний посібник для студентів, які навчаються
за спеціальністю 122 «Комп`ютерні науки та інформаційні технології»
спеціалізацією «Інформаційні технології в біології та медицині»

Київ

КПІ ім. Ігоря Сікорського

2018

Рецензенти: *Самсонов Валерій Васильович* к.т.н., проф

Вовянко Світлана Ігорівна., к.б.н., доцент

Відповідальний редактор: *Яковенко А.В., канд. техн. наук*

Гриф надано Методичною радою КПІ ім. Ігоря Сікорського (протокол № 10 від 21.06.2018р.)

за поданням Вченої ради факультету біомедичної інженерії (протокол № 10 від 30.05 18 р.)

Електронне мережне навчальне видання

Кисляк Сергій Володимирович

Настенко Євген Арнольдович

Основи молекулярної біології та біоінформатики

комп`ютерний практикум

Основи молекулярної біології та біоінформатики: комп`ютерний практикум [Електронний ресурс] : навч. посіб. для студ. спеціальності 122 «*Комп`ютерні науки та інформаційні технології спеціалізації «Інформаційні технології в біології та медицині»* / С.В.Кисляк, Є.А.Настенко; КПІ ім. Ігоря Сікорського. – Електронні текстові данні (1 файл, 2957 Кбайт). – Київ : КПІ ім. Ігоря Сікорського, 2018. – 95 с.

Навчальний посібник містить приклади використання сучасних програмних продуктів та інтернет-сервісів, що можуть бути використані для вирішення складних молекулярно-біологічних задач. На прикладі патогенного штаму E. Coli 0104:H4 висвітлені основні етапи проведення біоінформаційного аналізу біологічних послідовностей. Особливу увагу приділено алгоритмам вирівнювання біологічних послідовностей.

ЗМІСТ

ВСТУП

Предмет навчальної дисципліни «Біоінформатика» –- процес навчання та підготовки фахівця зі спеціальності 122 «Комп'ютерні науки та інформаційні технології» за спеціалізацією «Інформаційні технології в біології та медицині» першого (бакалаврського) рівня вищої освіти ступеня бакалавра, що дозволить використовувати методи молекулярної біології, генної інженерії, математичної статистики для задач аналізу та інтерпретації інформації, що міститься у високомолекулярних сполуках – біологічних полімерах. Навчальна дисципліна є основою для підготовки дипломних робіт за спеціальністю та в подальшій практичній роботі за фахом.

В структурно-логічній схемі програми підготовки фахівця:

- дисципліну **забезпечують** наступні дисципліни та кредитні модулі: Основи біології та медицини
- дисципліна **забезпечує** наступні навчальні дисципліни та кредитні модулі: аналіз біологічних послідовностей, моделювання систем

Метою навчальної дисципліни є формування у студентів здатностей:

- вирішувати проблеми в професійній діяльності на основі аналізу й синтезу;
- застосовувати сучасні парадигми програмування під час програмної реалізації професійних задач;
- обирати основні підходи, методи та інструментальні засоби для проведення та складання звітів з медико-біологічних досліджень, застосовувати сучасні методи та засоби аналізу даних та моделювання
- застосовувати та проводити аналіз великих наборів біологічних даних, за допомогою машинних алгоритмів та статистичних

методів, включаючи традиційні методи секвенування ДНК та конструювання сигнальних мереж за даними ДНК-мікрочіпів

Основні завдання навчальної дисципліни.

Згідно з вимогами освітньо-професійної програми студенти після засвоєння навчальної дисципліни мають продемонструвати такі результати навчання:

знання:

- математичних та природничих наук, в тому числі з інтелектуального аналізу даних, технологій видобування інформації
- інформаційних технологій, мов програмування, інструментарію програміста
- мов програмування, сучасних теорій організації баз даних та знань, методів і технологій їх розробки
- основних структур даних і алгоритмів
- загальних відомостей про організм людини і його функції з позицій системного підходу та використання їх в біомедичній кібернетиці
- фундаментальних основ фізичних та біологічних механізмів в організмі людини, що входять до основ біомедичних технологій

вміння:

- використовувати сучасні методи обробки та аналізу інформації стосовно об’єктів біологічної та медичної природи

- аналізувати великі набори біологічних даних за допомогою статистичних методів та алгоритмів машинного навчання;
- розвивати інженерне мислення та застосовувати нові підходи до вивчення, математичного опису та моделювання медико-біологічних об’єктів;

досвід в системі типових завдань діяльності:

- у вирішенні складних молекулярно-біологічних задач, пов`язаних з повним описом (анотуванням) біологічних послідовностей.

При виконанні комп’ютерних практикумів, студенти мають здобути навички роботи з базами даних біологічних послідовностей та основними біоінформаційними сервісами, використання яких дозволять провести детальний опис (анотування) амінокислотних та нуклеотидних послідовностей. Кожний етап аналізу, починаючи з секвенування, асемблювання послідовностей, картування прочитаних фрагментів (ресеквенування), визначення кодуючих ділянок ДНК (генів), передбачення просторової структури білків тощо, пов`язаний з вирішенням складних розрахункових задач. В основі вирішення кожної такої задачі лежить певний алгоритм. При виконання комп`ютерних практикумів першочерговим є аналіз застосованого алгоритму. Перед виконанням практикуму студент ознайомлюється з теоретичними відомостями відповідно до зазначеної теми, може проглянути приклад використання вказаного методу (алгоритму) для розв’язання поставленої задачі та проаналізувати одержані результати. Для самоконтролю студенти можуть скористуватись питаннями для самоконтролю, що забезпечить повторення та закріплення пройденого матеріалу.

Комп'ютерний практикум №1. Оцінка якості даних секвенування. Технологія очистки даних.

Мета роботи : ознайомитись з особливостями оцінки якості даних секвенування.

Основні задачі роботи:

1. Вміти оцінювати якість отриманих даних та навчитись проводити їх очистку
2. Ознайомитись з особливостями запису інформації про біологічні послідовності . Формати FastQ, Fasta.

Теоретичні відомості

Сучасні методи секвенування (наприклад Illumina, SOLID) біологічних послідовностей передбачають використання двох бібліотек фрагментів ДНК: «Paired-end» и «Mate-pair».

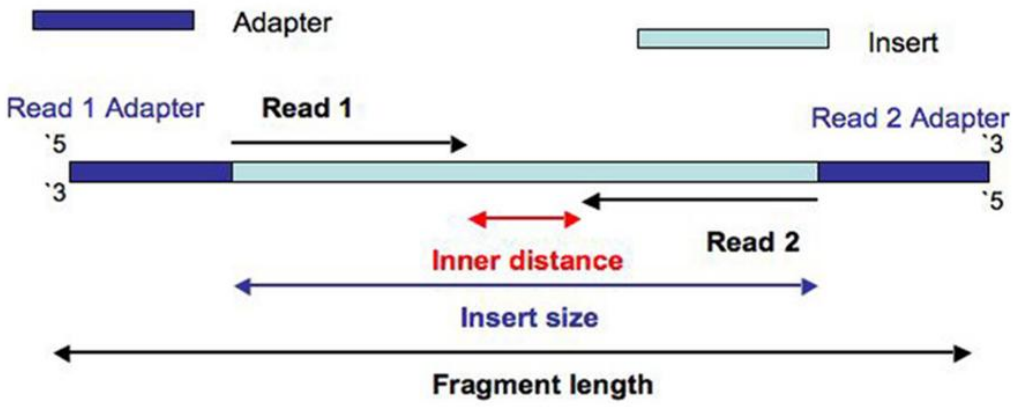


Рисунок 1 – «Paired-end» бібліотека Illumina. Відповідно до [1].

Процес створення «Paired-end» бібліотеки фрагментів ДНК для методів секвенування наступного покоління (NGS) вимагає виконання наступних кроків:

Фрагментації ДНК (ультразвукова або ферментативна фрагментація)

Лігування адаптерів. На даному етапі відбувається приєднання службових послідовностей – адаптерів, що дозволяють закріпити фрагмент ДНК до відповідної ділянки проточної комірки (flowcell) секвенатора.

Відбір фракцій фрагментів ДНК необхідної довжини.

На наступному етапі проводять *ампліфікацію* ДНК. Технології секвенування 454 Life Sciences компанії Roche, SOLID та Ion Proton використовують емульсійну ПЦР. Секвенатори компанії Illumina дозволяють отримати копії фрагментів за допомогою мостової ампліфікації.

Отримані фрагменти ДНК можуть бути прочитані з обох сторін у напрямку $3^{/}$ - $5^{/}$ кінця. Причому довжина таких прочитань (отримані короткі послідовності називають рідами (від. англ. reads)), зазвичай, не перевищує 200 пар нуклеотидів. Досить важливою характеристикою «Paired-end» бібліотеки є розмір вставки (insert size), що враховує довжину двох фрагментів ДНК, для яких отримані послідовності нуклеотидів, між якими розташована не прочитана послідовність нуклеотидів відомої довжини (inner size) (див. рисунок 1,2).

Процес синтезу «Mate-pair» бібліотеки для секвенування нетривіальний та проводиться відповідно до схеми (рисунок 2). На першому етапі фрагментують геномну ДНК. При цьому отримують фрагменти довжиною від 2kb до 40kb (фрагмент AB на рисунку 2). Потім проводять відбір фракцій фрагментів ДНК необхідної довжини. Після чого до фрагментів приєднують адаптери. Адаптери взаємодіють один з одним з формуванням кільцевої структури. Фрагментація циклічної ДНК, дозволяє отримати лінійні фрагменти довжиною від 200 до 600 нуклеотидів, в середній частині яких знаходяться адаптери та фрагменти, в яких адаптери

відсутні. Відбір фрагментів, що містять адаптери проводять за рахунок взаємодії біотильованих адаптерів з стрептавидином, що міститься на мікроносіях [2]. Після стандартної процедури лігування адаптерів проводять відбір фракцій необхідної довжини. Клональна ампліфікація є завершальним етапом синтезу «Mate-pair» бібліотек для секвенування. В порівнянні з «Paired-end» бібліотекою «Mate-pair» має значно більший розмір вставки (2-40kb). При цьому довжина парних рідів лімітована 36 нуклеотидами [2]. Прочитані фрагменти для «Mate-pair» бібліотеки мають протилежний напрямок (у напрямку від точки A та точки B фрагменту) у порівнянні з «Paired-end» (див. рисунок 2).

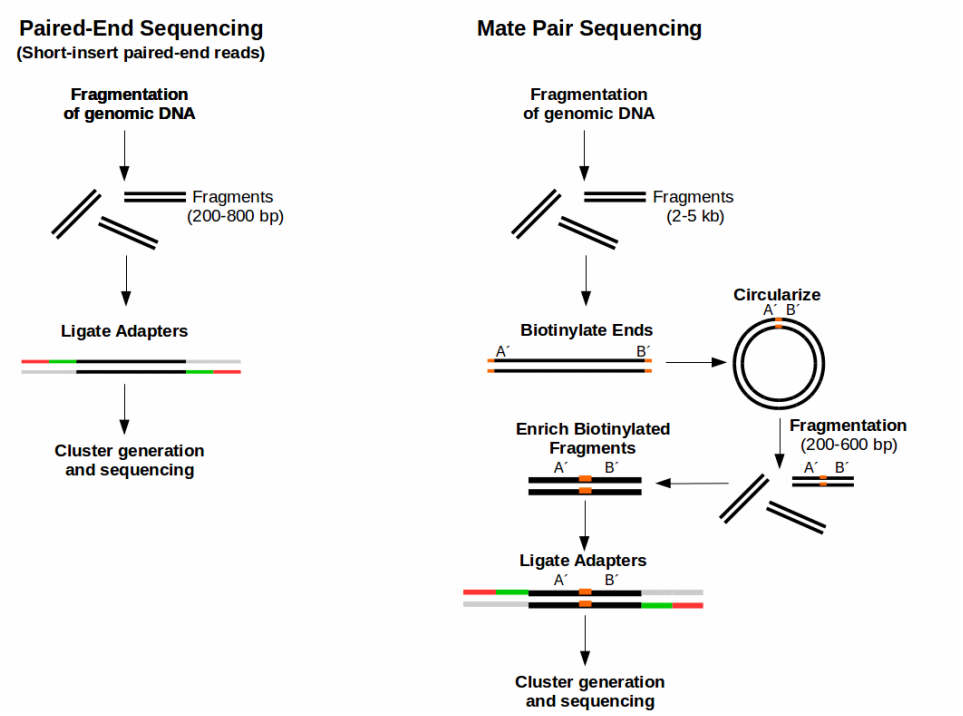


Рисунок 2 – Схема синтезу «Paired-end» та «Mate-pair» бібліотек для секвенування. Відповідно до [2]

Результати роботи секвенатора - це величезна кількість фрагментів ДНК, так званих «reads», з яких необхідно зібрати вихідну послідовність геному або його фрагменту. Як і будь-який прилад, секвенатор може

допускати помилки. Не всі нуклеотиди можуть бути прочитані вірно. Якщо ці помилки вчасно не виявити, то вони вплинуть на всі наступні етапи аналізу та, в кінцевому рахунку, геном буде зібраний з помилками. Враховуючи велику вартість прочитаного геному, на етапі асемблювання геному необхідно використовувати тільки «чисті» дані. Після того, як були отримали ріди на етапі секвенування, необхідно в першу чергу перевірити їх якість, виявити проблеми, якщо вони є, усунути їх та повторно перевірити якість. Циклічна перевірка даних виключає можливість використання даних низької якості. Якщо результат буде задовільним, переходимо до наступного етапу аналізу біологічних послідовностей. На якість отриманих даних можуть впливати:

1. *Ріди низької якості* містять нуклеотиди, які можуть бути помилково прочитані.

2. *Контамінації*, тобто домішки ДНК іншого організму. У зразок можуть потрапити бактерії з повітря, з поверхні рук експериментатора або при первинному відборі зразків туди може потрапити фрагмент тканини будь-якого іншої організму.

3. *Службові послідовності*. До службових відносять послідовності-адаптери, через які фрагменти ДНК кріпляться до поверхні проточної комірки (flow cell) або індекси, які дозволяють секвенувати за один прогін секвенатора відразу декілька геномів.

4. *Артефакти створення бібліотек*. Під час підготовки ДНК до секвенування на етапі полімеразної ланцюгової реакції можлива ситуація, коли деякі фрагменти виявляться представлені в кінцевій суміші в набагато більшій кількості копій, ніж інші.

5. *Використання різних форматів запису якості даних*. У різних приладів та різних версій програмного забезпечення формати даних відрізняються, тому таку особливість необхідно враховувати на етапі очистки даних секвенування.

В молекулярній біології та біоінформатиці найбільш популярним та розповсюдженим є FASTA формат. Він виглядає наступним чином: кожен запис в ньому складається з двох рядків. Перший рядок, який починається зі знака >, за ним може знаходитись інформація про базу даних в якій зберігається послідовність. Далі може знаходитись ідентифікатор або код доступу до послідовності. Ідентифікатор зазвичай містить три перших літери родової та дві літери видової таксономічної приналежності організму, для якої була отримана послідовність. Після службової інформації в наступному рядку безпосередньо записана сама послідовність. FASTA формат дозволяє зберігати інформацію для нуклеотидних та амінокислотних послідовностей. Наприклад, послідовність микротубулярного протеїну людини у FASTA форматі має наступний вигляд:

>gi|31563518|ref|NP_852610.1| microtubule-associated proteins 1A/1B light chain 3A isoform b [Homo sapiens]
MKMRFFSSPCGKAAVDPADRCKEVQQIRDQHPSKIPVIIERYKGEKQLPVLDKTKFLVPDHVNMSE
LVKIIRRRLQLNPTQAFFLLVNQHSMVSVSTPIADIYEQEKDEDGFLYMVYASQETFGFIRENE

Розглянемо організацію файлу з даними на прикладі найбільш поширеного формату секвенування – FASTQ. Для кожного ріда запис у форматі FASTQ містить 4 рядка та виглядає наступним чином:

@SEQ_ID
GATTTGGGGTTCAAAGCAGTATCGATCAAATAGTAAATCCATTTGTTCAACTCACAGTTT
+

!''((((***+))%%%++)(%%%%).1***-+*''))**55CCF>>>>>>CCCCCCC65*

Перший рядок починається значком @ та містить інформацію про назву послідовності (інколи записують інформацію про довжину послідовності). Другий рядок - це нуклеотидна послідовність прочитання. Третій рядок містить знак + та зазвичай залишається пустим. В четвертому рядку для кожного з прочитаних нуклеотидів записано значення якості. Зазвичай для запису інформації про якість прочитань (рідів) використовують значення Phred, що можна визначити по формулі:

$$\boldsymbol{Q = -10log_{10}P}$$

де Q- значення Phred, P - вірогідність помилки

Для даної шкали кожному нуклеотиду відповідає значення, яке логарифмічно залежить від імовірності помилки. Розрахунок параметру $\boldsymbol{P}$ проводять за допомогою секвенування відомої нуклотидної послідовності. Тобто, наприклад, якщо у нас є послідовність, в якій для всіх нуклеотидів значення дорівнює 10, це значить, що у нас в середньому зустрічається 1 помилка на кожні 10 нуклеотидів. Якщо значення Phred дорівнює 20 - це 1 помилка на кожні 100 нуклеотидів, якщо 30 - 1 на 1000 і так далі. Для отримання якісної збірки геному необхідно працювати з даними, якість яких буде не нижче 30. Значення Q – це, як правило, двозначне число. Для економії пом`яті зручно записувати кожне значення Q у вигляді одного символу, що буде розташований навпроти кожного нуклеотиду у четвертому рядку FASTQ файла. Ця проблема вирішується таким чином: в комп'ютерах будь-який символ, цифру або значок можна представити у вигляді певного коду, який записаний в таблиці ASCII (American Standard Code for Information Interchange). Перші 32 символи (див. таблиця 1) в цій таблиці - це пробіл, табуляція та інші символи, які записати неможливо. Тому для запису використовують шкалу Phred 33, в якій відлік

починається з 33-го символу, тобто 33-й символ ми вважаємо першим, тому якість 1 позначається знаком «!». Якщо потрібно записати якість 30, то ми беремо 63-й символ, що позначається знаком «?».

Таблиця 1 – Таблиця ASCII

Dec	Hex	Name	Char	Ctrl-char	Dec	Hex	Char	Dec	Hex	Char	Dec	Hex	Char
0	0	Null	NUL	CTRL-@	32	20	Space	64	40	@	96	60	`
1	1	Start of heading	SOH	CTRL-A	33	21	!	65	41	A	97	61	a
2	2	Start of text	STX	CTRL-B	34	22	"	66	42	B	98	62	b
3	3	End of text	ETX	CTRL-C	35	23	#	67	43	C	99	63	c
4	4	End of xmit	EOT	CTRL-D	36	24	$	68	44	D	100	64	d
5	5	Enquiry	ENQ	CTRL-E	37	25	%	69	45	E	101	65	e
6	6	Acknowledge	ACK	CTRL-F	38	26	&	70	46	F	102	66	f
7	7	Bell	BEL	CTRL-G	39	27	'	71	47	G	103	67	g
8	8	Backspace	BS	CTRL-H	40	28	(	72	48	H	104	68	h
9	9	Horizontal tab	HT	CTRL-I	41	29	)	73	49	I	105	69	i
10	0A	Line feed	LF	CTRL-J	42	2A	*	74	4A	J	106	6A	j
11	0B	Vertical tab	VT	CTRL-K	43	2B	+	75	4B	K	107	6B	k
12	0C	Form feed	FF	CTRL-L	44	2C	,	76	4C	L	108	6C	l
13	0D	Carriage feed	CR	CTRL-M	45	2D	-	77	4D	M	109	6D	m
14	0E	Shift out	SO	CTRL-N	46	2E	.	78	4E	N	110	6E	n
15	0F	Shift in	SI	CTRL-O	47	2F	/	79	4F	O	111	6F	o
16	10	Data line escape	DLE	CTRL-P	48	30	0	80	50	P	112	70	p
17	11	Device control 1	DC1	CTRL-Q	49	31	1	81	51	Q	113	71	q
18	12	Device control 2	DC2	CTRL-R	50	32	2	82	52	R	114	72	r
19	13	Device control 3	DC3	CTRL-S	51	33	3	83	53	S	115	73	s
20	14	Device control 4	DC4	CTRL-T	52	34	4	84	54	T	116	74	t
21	15	Neg acknowledge	NAK	CTRL-U	53	35	5	85	55	U	117	75	u
22	16	Synchronous idle	SYN	CTRL-V	54	36	6	86	56	V	118	76	v
23	17	End of xmit block	ETB	CTRL-W	55	37	7	87	57	W	119	77	w
24	18	Cancel	CAN	CTRL-X	56	38	8	88	58	X	120	78	x
25	19	End of medium	EM	CTRL-Y	57	39	9	89	59	Y	121	79	y
26	1A	Substitute	SUB	CTRL-Z	58	3A	:	90	5A	Z	122	7A	z
27	1B	Escape	ESC	CTRL-[	59	3B	;	91	5B	[	123	7B	{
28	1C	File separator	FS	CTRL-\	60	3C	<	92	5C	\	124	7C	|
29	1D	Group separator	GS	CTRL-]	61	3D	=	93	5D	]	125	7D	}
30	1E	Record separator	RS	CTRL-^	62	3E	>	94	5E	^	126	7E	~
31	1F	Unit separator	US	CTRL-_	63	3F	?	95	5F	_	127	7F	DEL

У деяких версіях програмного забезпечення секванаторів наступного покоління (наприклад Illumina) зустрічаються інша шкала якості, в якій відлік йде з 64-го символу (шкала Phred 64). Тому досліднику важливо знати, з якою саме шкалою якості він працює, тому що один і той самий символ в різних шкалах може означати зовсім різні значення якості.

Одним з найбільш популярних інструментів для перевірки якості даних секвенування є FastQC . Програмне забезпечення FastQC бере на вхід вихідні дані секвенатора у форматі FASTQ та створює звіт в форматі html. Завантажити програму можна за посиланням

https://www.bioinformatics.babraham.ac.uk/projects/fastqc/. Звіт програми, містить досить багато різних графіків, кожен з яких несе свою інформацію. Проведемо аналіз рідів на прикладі дуже поганого та якісного набору даних. Перш за все програма видає нам загальну інформацію: це назва файлу, формат, запис та якість, кількість послідовностей та їх довжина, а також GC-склад (рисунок 3).

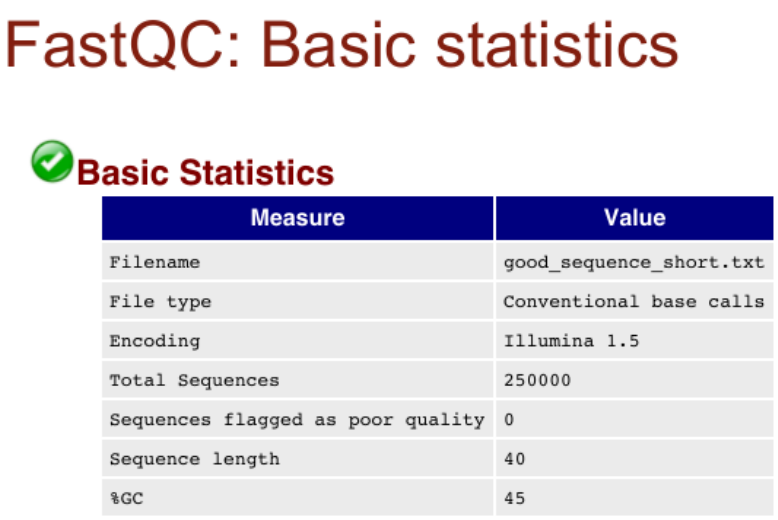


Рисунок 3- Загальна інформація звіту FastQC [3]

Найбільш корисним є графік розподілу якості впродовж рідів (див. рисунок 4). По вісі x позначається позиція нуклеотиду ріда, а по вісі Y – середня якість прочитаного нуклеотида для відповідної позиції. На графіку ми бачимо, що на самому початку з $5^{/}$ кінця для всіх рідів якість досить висока. Поступово, ближче до $3^{/}$ кінця якість починає падати (див. рисунок 1 bad). Неякісні дані необхідно «почистити». В такій ситуації необхідним є видалення фрагменту нуклеотидів з $5^{/}$ кінця починаючи приблизно з 20 нуклеотиду. Після такої операції дані будуть більш якісними, хоча довжина рідів зменшиться, що може вплинути на якість зібраного геному.

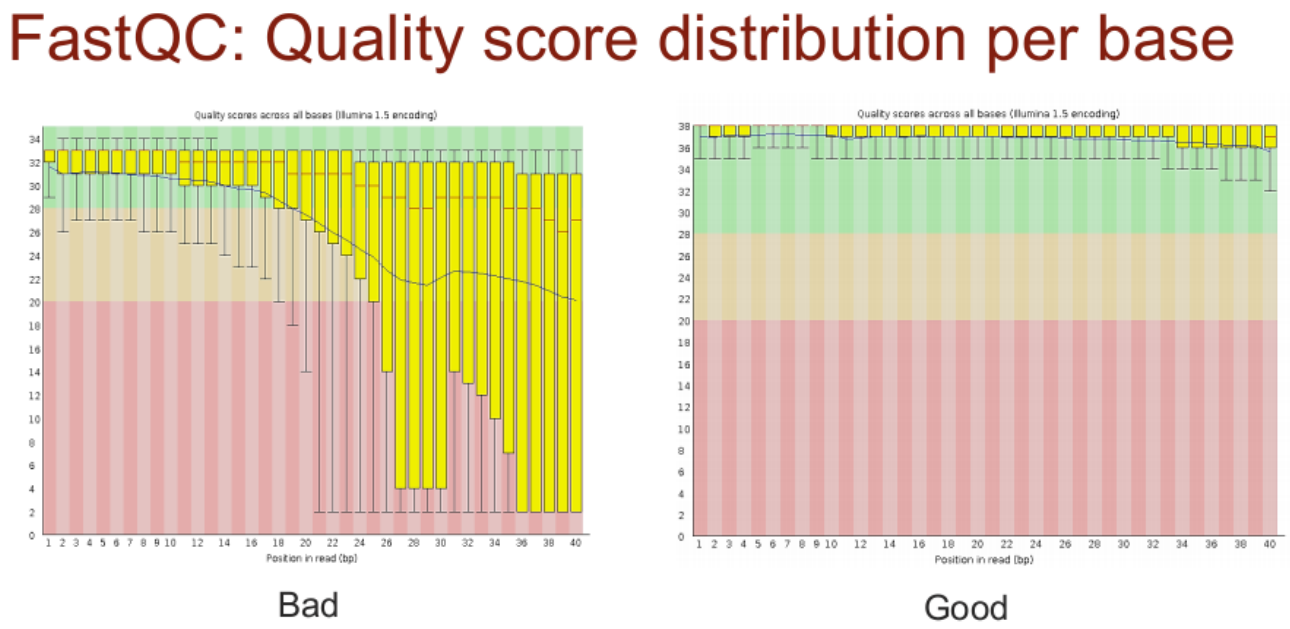


Рисунок 4- Графік розподілу якості впродовж рідів [3]

На наступному графіку (див. рисунок 5) показано розподіл якості по рідам. Для кожного позиції ріда розраховується середнє значення якості (вісь x) та підраховують кількість таких рідів.

Рисунок 5 – Графік розподілу якості по рідам [3]

На наступному графіку (див. рисунок 6) показано розподіл чотирьох гетероциклічних основ (A,T,G,C) в рідах. Якщо б ріди розташовувалися випадково, то цей розподіл був би рівномірним (рисунок 4 good).

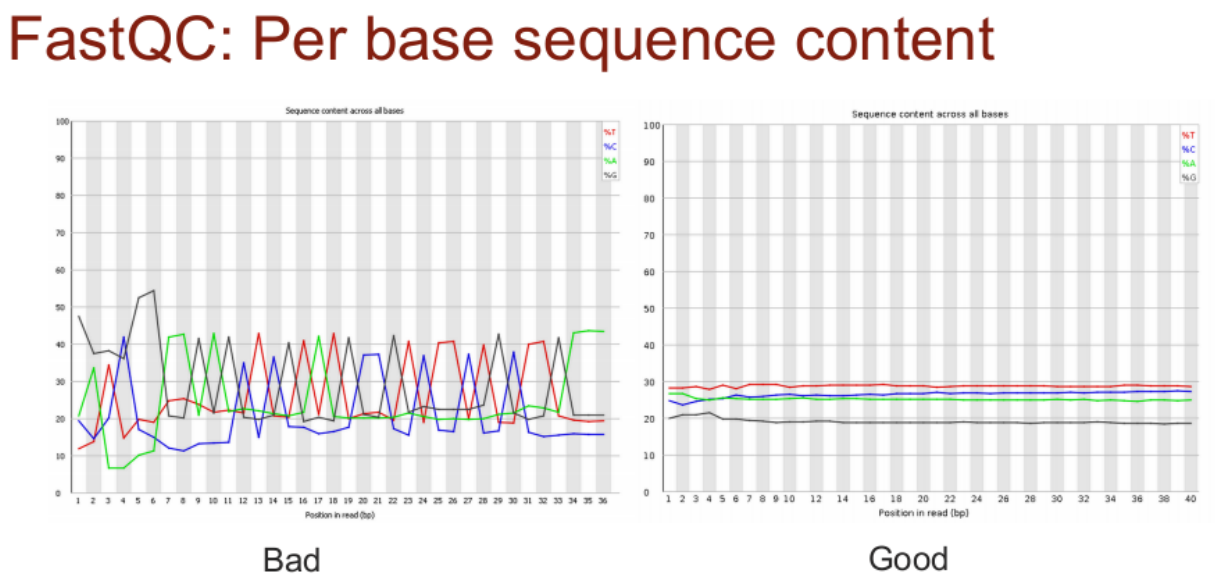


Рисунок 6 – Графік розподілу гетероциклічних основ [3]

Якщо у рідах зустрічається досить часто один і той самий фрагмент це може вказувати на наявність службової послідовності (див. рисунок 6 bad). Наступний графік - це розподіл рідів по їх середньому GC складу. Ріди, що були отримані з одного генома, повинні мати приблизно однаковий GC склад. В геномах зустрічаються більш та менш багаті GC ділянки, тому ми маємо не вузький пік, а деякий розподіл (див. рисунок 7 good). Декілька піків розподілу може вказувати на присутність контамінацій (див. рисунок 7 bad)

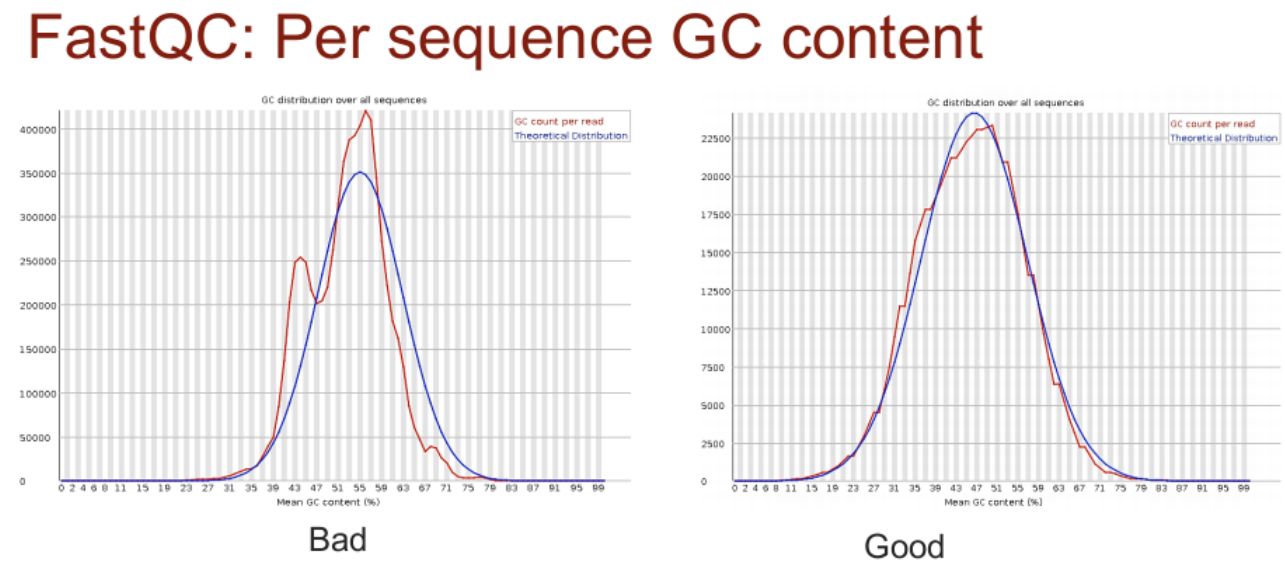


Рисунок 7- Графік розподілу рідів по GC-складу [3]

На графіку (рисунок 6) представлений рівень дуплікації наших рідів. Тобто дивлячись на нього ми можемо побачити, скільки раз зустрічається той чи інший фрагмент.

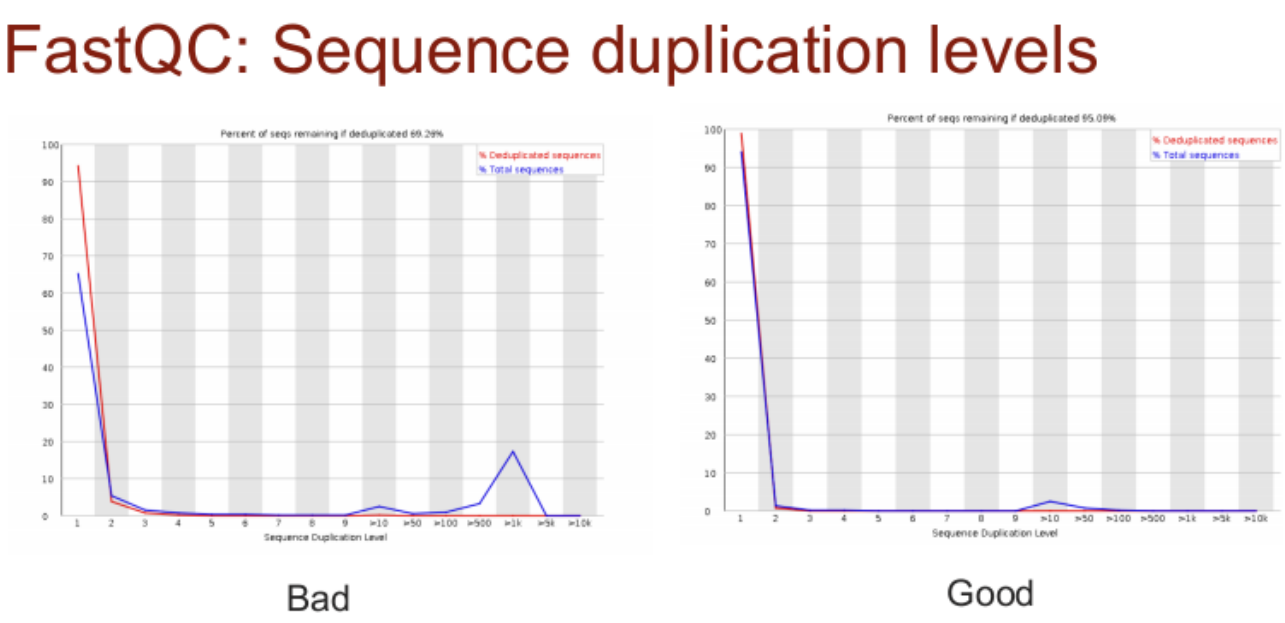


Рисунок 8 – Графік розподілу дуплікації рідів [1]

Якщо створена бібліотека для секвенування послідовностей враховувала, наприклад тридцятикратне покриття, то кожен фрагмент буде

зустрічатися з частотою близько 30. На рисунку 8, зліва ми бачимо, що у нас є близько 10% фрагментів, які зустрічаються більше ніж 1000 разів (це вказує на службові послідовності, які необхідно видалити). В наступній частині звіту представлені, ті послідовності, які зустрічаються частіше, ніж очікується, та висновки про наявність яких вже були зроблені за попередніми результатами (рисунок 7) FastQC дає змогу побачити список послідовностей, відсоток їх появи та назви службових послідовностей, що залишились у рідах. Таким чином ми можемо визначити присутність адаптерів, індексів, праймерів та інших службових послідовностей, які використовуються на етапі створення бібліотек для секвенування. Після ідентифікації службових послідовностей наступним етапом буде їх видалення.

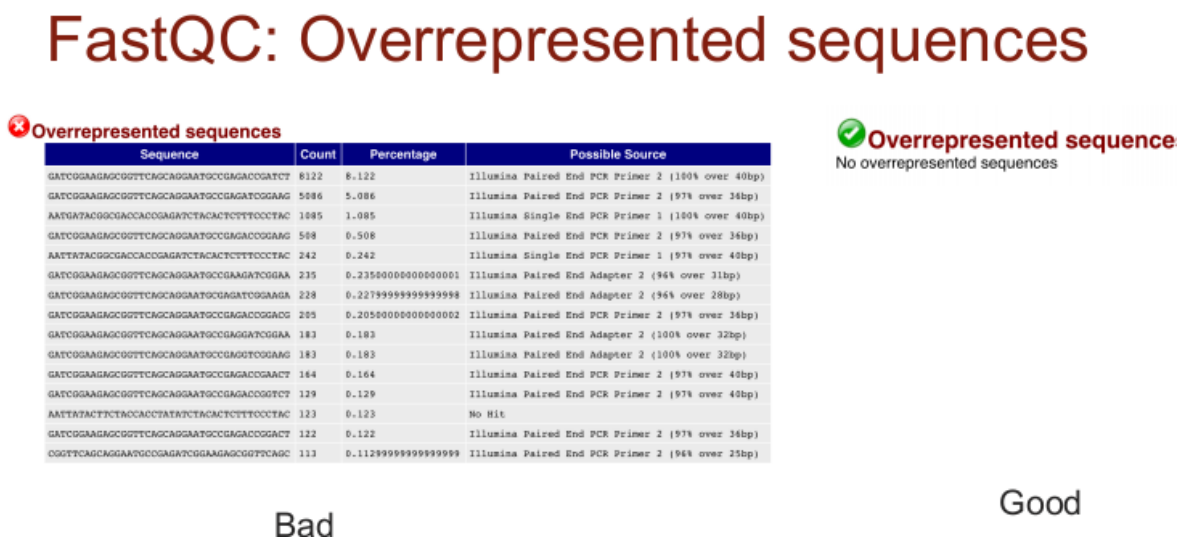


Рисунок 9 – Службові послідовності [3]

Порядок виконання роботи

Деякі штами кишкової палички (*Escherichia coli*) є патогенними. Один з таких штамів, кишкової палички O104:H4, став причиною спалаху харчових отруєнь в Європі в 2011році. Необхідно провести геномний аналіз даного штаму E.coli, щоб з'ясувати причину патогенності. Для

цього, на першому етапі біоінформаційного аналізу, перевіримо якість отриманих даних (платформа для секвенування - Illumina), після процедури очистки, необхідно асемблювати бактеріальний геном патогенного штаму, проанотувати його та порівняти з референсними геномом [4-6].

- Для виконання роботи необхідно скачати парні ріди, що отримані для патогеннго бактеріального організму *Escherichia coli* O104:H4 [7]. Якість отриманих даних оцінюємо окремо для прямих та зворотних рідів. Скільки рядків у кожному з цих двох файлів ? Число рядків в Linux можна підрахувати командою *wc –l* < ім'я файлу >). За допомогою команди cat < ім'я файлу > можна побачити вміст файлу. Використовувати команду cat необхідно для розархівованого файлу.
- Завантажте на комп'ютер програму FastQC. За допомогою програми *проаналізуйте* файли з рідами. Яким чином можна покращити їх якість? Скільки пар ридів (прямі, зворотні) в цих даних ? Яку довжину нуклеотидів має рід? Яка шкала Phred (33, 64) використовується для цих даних? Чому значення кількості рідів (програма FastQC), не збігається з значенням кількісті рядків файлу (terminal linux) ?
- Наступний етап біоінформаційного аналізу пов`язаний з покращенням якості рідів. Для таких задач будемо використовувати програму Trimmomatic, що запускається у консольному режимі операційної системи Linux. При цьому рекомендується створити папку, наприклад genome project, в якій необхідно зберегти інсталяційний пакет програми Trimmomatic та парні ріди у форматі FastQ. Встановлення програми здійснюють за допомогою команд java –jar та вказують відповідний шлях до інсталяційного файлу. Перед запуском програми необхідно обновити системні файли (команда sudo apt-get update) та

перевірити чи встановлена Java Runtime Environment (команда sudo apt-get install default-jre).

- Після запуску програми без відповідних аргументів, програма видає підказку, в якій представлена інформація, яким чином її потрібно запускати (рисунок 10)

```
java -jar <path to trimmomatic.jar> PE [-threads <threads] [-phred33 | -phred64] [-trimlog
<logFile>] >] [-basein <inputBase> | <input 1> <input 2>] [-baseout <outputBase> |
<unpaired output 1> <paired output 2> <unpaired output 2> <step 1> ...
```

Рисунок 10– Інформаційне супроводження першого запуску програми Trimmomatic для парних рідів

Спочатку вказують, що ріди парні **PE** (ми працюємо з парними рідами, що отримані за допомогою платформи Illumina). На наступному кроці необхідно вказати шкалу оцінки якості рідів **–phred33** або **- phred 64**. Потім записують вихідні дані (назву файлів з прямими та зворотними рідами) **SRR292770_1.fastq SRR292770_2.fastq.** Наступна інформація стосується файлів, що будуть зберігати інформацію по *прямим рідам*, які мають пару (наприклад **forvardPe_1.fq**) та прямим рідам, для яких пара була втрачена, що пов`язано з низькою якістю ріда (**singlePe_1.fq**). Теж саме необхідно зробити для *зворотних рідів* (**reversePe_2.fq** та **singlePe_2.fq).** Далі ми записуємо команди, що дозволяють нам фільтрувати та чистити отримані дані, а нам необхідно:

- відрізати з початку та з кінця кожного ріда нуклеотиди з якістю нижче 30;
- відкинути всі ріди, що мають довжину менше 30 нуклеотидів. Скільки пар рідів залишилося після таких операцій ? Для виконання завдання необхідно ознайомитись з інструкцією користувача програми Trimmomatic

http://www.usadellab.org/cms/uploads/supplementary/Trimmomatic/TrimmomaticManual_V0.32.pdf [4].

Питання для самоконтролю:

1. Метод секвенування першого покоління. Метод Sangera ?
2. Особливості проведення та основні компоненти полімеразної ланцюгової реакції. Види ПЛР ?
3. Особливості створення бібліотек для секвенування біологічних послідовностей. Paired end Mate pair бібліотеки ?
4. Методи секвенування наступного покоління. Платформи Illumina, Roche 454, Ion Torrent ?
5. Методи секвенування одиничних молекул. Платформа Pacific biosciences, Oxford nanopore ?
6. Методи оцінки та покращення якості рідів. Формат FastQ ?

Комп'ютерний практикум №2 Алгоритми асемблювання геномів. Оцінка якості отриманої збірки геному

Мета роботи: навчитись асемблювати геноми про- та еукаріотичних організмів та оцінювати якість отриманої збірки.

Основні задачі роботи:

1. Ознайомитись з алгоритмом Overlap layout consensus (OLC)
2. Ознайомитись з алгоритмом збірки геномів на основі графів де Брейна.

Теоретичні відомості

На сьогоднішній день для задач асемблювання геномів використовують два основних алгоритма [8]. Перший з них був

запропонований у 1980 році та отримав назву Overlap layout consensus. Це інтуїтивно зрозумілий алгоритми, який передбачає спочатку попарне порівняння та вирівнювання усіх рідів. Після чого будується граф, в якому вузлами є ріди. Якщо перекриття між рідами виявляється не менше, ніж задане порогове значення T, вузлу прописують зв'язок (ребро).

Наприклад, якщо у нас є ріди ACGT, CGTT, TTAA и нам потрібно зібрати фрагмент геному при пороговому значенні T=2. Після проведення вирівнювання ми бачимо, що ріди перекриваються. Побудуємо граф, в якому ребра будуть поєднувати вершини графу (ріди) за умов перекриття. Граф буде мати вигляд ACGT→CGTT→TTAA. Задача встановлення первинної послідовності нуклеотидів зводиться до знаходження загального підрядка для заданих рядків. Гамільтонів цикл проходить по кожній вершині графа рівно один раз. Такий шлях по графу дозволяє визначити послідовність нуклеотидів геному. Для даного прикладу послідовність фрагменту геному буде мати вигляд ACGTTAA.

При використанні алгоритму OLC кількість контигів (неперервні послідовності, що були отримані при перекритті рідів) оцінюють відповідно до співвідношення [9]:

$$n = Nexp\ \ -c\frac{Lr-T+1}{Lr}\ ,$$

де $c = NLr/Lg$ – глибина секвенування (покриття геному рідами), $N-$ кількість рідів, $Lr-$ середня довжина ріда $,Lg-$довжина геному, $T-$ порогове значення перекриття рідів.

OLC алгоритм збірки геному використовуються багатьма програмами-асемблерами, такими як: Arachne, CAP3, Phusion, Celera Assembler.

Алгоритм OLC використовують для рідів великої довжини (метод Sangera, Pacific Biocinses). Для N рідів необхідно порівняти кожний рід с

кожним, тобто необхідно виконати $N(N-1)\sim N^2$ порівнянь. З появою методів секвенування наступного покоління довжина ріда зменшилась з відповідним зростанням кількості рідів. Задачу знаходження шляху по вершинам графу можна вирішити за експоненційний час, що є недопустимим, враховуючи велику кількість вихідних даних секвенаторів NGS (наприклад Illumina).

Для NGS використовується інший алгоритм збірки геномів. Алгоритм базується на використанні графів де Брейна. Для кожного ріда формується префікс та суфікс однакової довжини k ($k-$мери). Кожний рід в структурі графа містить дві вершини, що поєднується ребром. Ребро – це $k+1-$ мер, що відображає певний рід. Вершини поєднуються в структурі графа, якщо префікс відповідає суфіксу $k-$мера. Знайдений Ейлерів цикл в графі дозволяє отримати послідовність нуклеотидів геному. Складність такого алгоритму $O(N)$. Такий лінійний алгоритм можливо ефективно використовувати для вихідних даних секвенаторів NGS.

На рисунку 11 представлений орієнтовний збалансований граф для k=2.

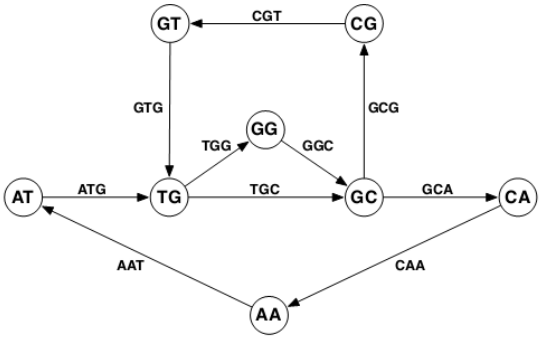


Рисунок 11 – Приклад графа де Брейна для k=2 [10]

Знайдений ейлерів цикл в графі дозволить відновити послідовність нуклеотидів геному. Для нашого прикладу такий кільцевій фрагмент геному сформований послідовністю нуклеотидів ATGGCGTGCA.

Кількість контигів в даному алгоритмі можна розрахувати відповідно до співвідношення:

$$n = Nexp(-d_k),$$

де $d_k = \frac{N(Lr-K+1)}{Lg}$, N −кількість рідів, Lr − середня довжина ріда, Lg −довжина геному, K − довжина k − мера.

Для оцінки якості отриманих збірок геномів у вигляді набору контигів та скефолдів необхідно враховувати:

- Кількість отриманих контигів
- Загальну довжину контигів
- Довжина найбільшого контигу
- GC склад
- Одним з основних параметрів є N50 – це такий розмір контига, після якого інші контиги складають не менше 50% довжини генома.
- L50 – значення сумарної кількості контигів, що відповідає не менше 50% довжини генома

Порядок виконання роботи

Після ітеративного процесу очистки даних секвенування необхідно отримати автоматичну збірку (Standard draft) геному *Escherichia coli* у вигляді набору скафолдів різної довжини [11]. Для задач збірки геномів використовують спеціальні програми - геномні асемблери.

Для автоматичної оцінки якості отриманих збірок та перевірки результатів роботи геномного асемблера (Standard draft геному був отриманий за допомогою SPAdes асемблера) можна використати програму QUAST з веб -інтерфейсом (http://quast.bioinf.spbau.ru/). Також реалізована можливість запуску програми в терміналі Linux [4-6].

Відповідно до триманих наборів скафолдів (scaffolds.fasta) та контигів (contigs.fasta) за допомогою програми QUAST необхідно визначити:

- Загальну довжину скафолдів та контигів
- Максимальну довжину контигів та скефолдів
- Загальну довжину скефолдів, враховуючи фрагменти довжиною менше 500 нуклеотидів
- N50 для скафолдів
- L50 для скафолдів
- Загальну кількість пропусків (гепів) для скафолдів
- Кількість пропусків на 30 kb

Проаналізуйте графіки оцінки довжин контигів (Cumulative length та Nx) та GC склад та дайте відповідь на запитання:

- Яку довжину має перший та останній контиг ?
- Визначити сусідні скафолди, для яких розбіжність по довжині буде максимальною, в порівнянні з іншими скафолдами ?
- Чому деякі значення (наприклад total length >=1000) для контигів та скафолдів не збігаються ?

Питання до самоконтролю:

1. Алгоритм асемблювання рідів з довжиною до 10 000 нуклеотидів?
2. Алгоритм асемблювання рідів для вихідних даних секвенаторів NGS?
3. Оцінка якості отриманої збірки геному? Параметри L50 та N50 ?
4. Алгоритм переходу від кільцевих до лінійних геномів ?
5. Яким чином в графі де Брейна враховуються повтори ?
6. За яких умов в графі буде виникати Ейлерів цикл? Теорема Ейлера.

Комп'ютерний практикум №3 Пошук гомологічних послідовностей за допомогою алгоритму BLAST

Мета роботи: використовуючи алгоритм пошуку схожих послідовностей BLAST (Basic Local Alignment Search Tool) знайти гіпотетичних гомологів відповідного білка.

Основні задачі роботи:

1. Ознайомитись з особливостями алгоритмів, що використовують хешування. Хеш-функція. Алгоритм BLAST.

2. Ознайомитися з основними принципами побудови матриць замін амінокислот PAM та Blosum.

Теоретичні відомості

Надзвичайно складний процес, що пов`язаний з описом біологічних послідовностей можна спростити, використовуючи, наприклад, алгоритми парного вирівнювання. Якщо для відповідної амінокислотної послідовності існують експериментальні дані та відомі її функції, просторова структура, особливості молекулярної організації гену, що кодує білок тощо (тобто послідовність є проанотованою), та існує друга послідовність, що є подібною до описаної, ми можемо припустити, що ці макромолекули виконують схожі функції. Вирівнювання біологічних послідовностей дозволяє частково або повністю перенести анотацію з описаної послідовності на неописану. При проведенні біоінформаційного аналізу на самому початку дослідження, коли відсутня інформація про досліджувану послідовність, виникає необхідність проведення порівняння даної послідовності з мільйонами послідовностей, що зберігаються в відомих базах даних (наприклад Uniprot для білкових послідовностей або Genbank для нуклеотидних послідовностей). Кількість можливих

вирівнювань двох послідовностей, що мають однакову довжину n, можна визначити за формулою:

$$N_{align} = C_{2n}^{n} = \frac{(2n)!}{(n!)^2} \approx \frac{2^{2n}}{\sqrt{\pi n}}$$

Якщо вважати, що середня довжина послідовності для, якої шукають гомолога має розмір ~ 1000, кількість варіантів вирівнювання буде надзвичайно великою, більше ніж уявна кількість атомів всесвіту ($\sim 4 \cdot 10^{81}$) Яким чином серед такої кількості варіантів знайти оптимальне вирівнювання ? Найбільш ефективний підхід у вирішенні задачі пошуку оптимального парного вирівнювання реалізований за допомогою методу динамічного програмування. Застосування таких алгоритмів (глобального та локального вирівнювання) має суттєві обмеження, що пов`язані з їх складністю. Використовуючи алгоритми парного вирівнювання неможливо ефективно вирішити задачу, що пов`язана з пошуком послідовності-гомолога, в базах даних амінокислотних або нуклеотидних послідовностей. Так, наприклад, необхідно порівняти деяка послідовність з послідовностями в банку даних розміром $L_1 = 10^8$ (розмір генома людини $3x10^9$ пар нуклеотидів). Розмір послідовності, для якої шукають гомолога, зазвичай, має розмір $L_2 = 10^3$ нуклеотидів. При порівнянні послідовностей на кожному кроці необхідно виконати 10^2 операцій. Якщо врахувати швидкодію звичайного комп'ютера ($\sim 10^9$операцій в секунду), тоді час, необхідний для пошуку оптимального вирівнювання однієї послідовності в БД навіть такого невеликого розміру, буде складати $10^6 сек = 11 днів$. Якщо просеквенований бактеріальний геном з, наприклад, 3000 генів (приблизно за тиждень), на те, щоб його проанотувати, буде витрачено $11 \cdot 3000 = 33000 днів$. Аналіз генома вимагає значно більше часу, ніж його секвенування [12]. Для вирішення задачі пошуку послідовності – гомолога в сучасних молекулярно-біологічних БД неефективно застосовувати

алгоритми парного вирівнювання. Розв'язок цієї проблеми полягає в тому, щоб ще до застосування *методів динамічного програмування* відібрати з БД кандидатів для порівняння. Ця ідея реалізована в програмі BLAST. Основна стратегія її роботи полягає в хешуванні. BLAST – алгоритм, що використовують для знаходження ділянок локальної подібності між послідовностями. Алгоритм порівнює вхідну послідовність (Query) з послідовностями у базах даних, шукає подібні послідовності та оцінює статистичну значущість знахідок.

Після введення запиту, BLAST створює хеш-таблицю (таблиця 2) списку слів (при повному збігу символів) та подібних слів (не повний збіг, при якому один або декілька символів відрізняються один від одного) довжини k для запиту (зазвичай 11 символів для нуклеотидних послідовностей та 3 - для амінокислотних). Слова (фрагменти біологічних послідовностей) записуються з кроком 1 та усіма можливими перекриттями (рисунок 12).

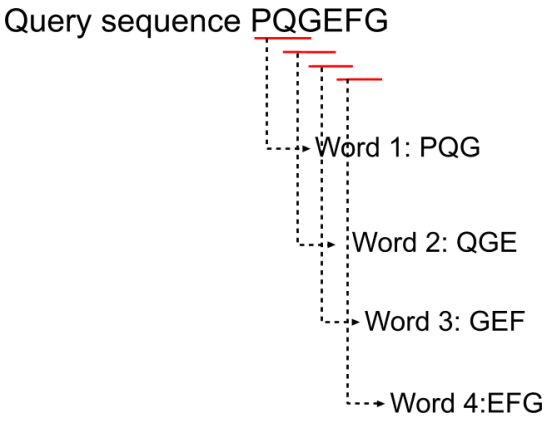


Рисунок 12 - Підрядки рядку запиту (PQGEFG) алгоритму Blast

Таблиця 2 – Хеш-таблиця списку слів довжиною 3

Слово	Позиція в банку даних
PQG	234, 687, 65444, …
QGE	346,65445,11235…

GEF	569, 987, 65446…
EFG	659, 20065,65447…

На наступному етапі проводиться відбір підрядків, що мають велику вагу вирівнювання з рядком запиту. При визначенні ваги вирівнювання використовують матриці замін амінокислот або застосовують схеми оцінок нуклеотидних замін. У більшості програм серії BLAST за замовчуванням використовується матриця BLOSUM 62. Матриця BLOSUM 62 отримана для гомологічних послідовностей, що мають 62% ідентичності.

Пошук починається з підрядка, що з максимальною вагою вирівнюється з рядком запиту. Відбувається пошук слів (подібних слів) в БД NCBI в порядку зменшення ваги вирівнювання.

Після того, як перший збіг знайдено, послідовність розширюється в обидві сторони по одному символу, до тих пір поки ступінь збігу буде не менше встановленого порогового значення або не закінчиться одна з послідовностей. За допомогою модифікованого алгоритму Сміта-Уотермана визначаються всі пари підрядків (запит (Query) - результат пошуку (Sbjct)), що вирівнюються з максимальною вагою (рисунок 13).

Остання стадія алгоритму пов`язана з оцінкою статистичної значущості результатів вирівнювання, що пов`язана з таким параметром як E-value. E-value - визначає ймовірність того, що вирівнювання не краще ніж випадкове (якщо послідовність запиту та банк даних були б заповнені випадково). Зрозуміло, що значення E-value повинно бути найменшим при максимальному відсотку ідентичності знайденої послідовності та послідовності запиту. Однак, якщо E-value порядку 10^{-3} чи більше, то, як правило, відповідність порівнювальних послідовностей носить, скоріше за все, випадковий характер. Якщо ж E-value менше за 10^{-3}, то такий результат пошуку є значущим та несе певну біологічну інформацію.

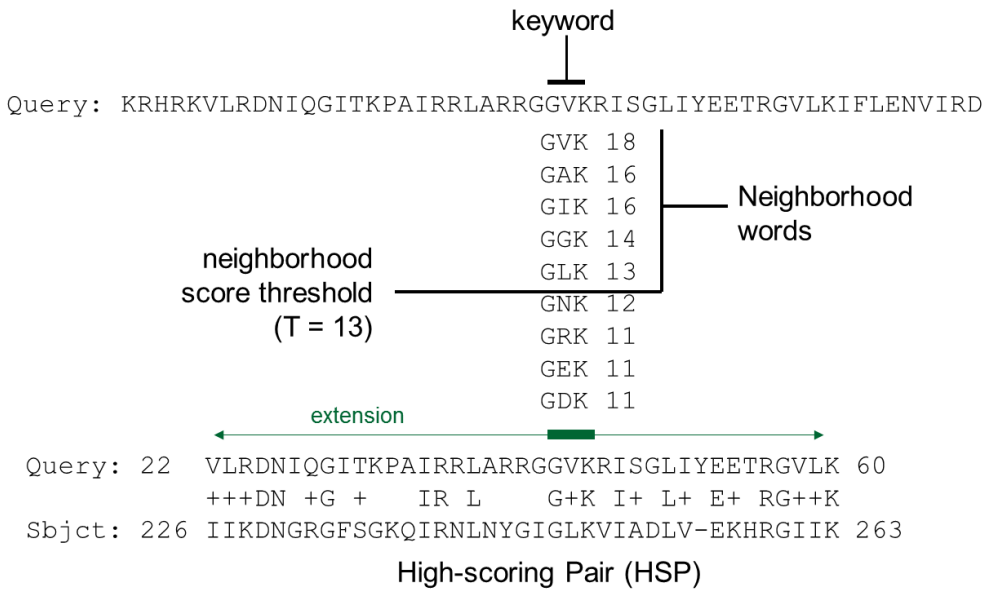


Рисунок 13 - Результат роботи алгоритму для нуклеотидної послідовності

Модифікації алгоритму BLAST

Nucleotide blast – пошук даної послідовності нуклеотидів в базах даних нуклеїнових кислот

Protein blast - пошук амінокислотної послідовності в БД амінокислотних послідовностей білків;

Blastx - переводить нуклеотидную послідовність в амінокислотну, а потім порівнює її з базою даних амінокислотних послідовностей білків;

Tblastn - амінокислотна послідовність порівнюється з трансльованими послідовностями бази даних нуклеїнових кислот;

tblastx - переводить нуклеотидну послідовність в амінокислотну, а потім порівнює її з трансльованою послідовністю нуклеїнових кислот.

Порядок виконання роботи

При виконанні завдань користуйтеся web-інтерфейсом до BLASTP на сервері NCBI: http://blast.ncbi.nlm.nih.gov/, далі в розділі Basic BLAST перейти за гіперпосиланням protein blast [13].

1. Пошук гіпотетичних гомологів досліджуваного білка в різних базах даних.

- Подайте на вхід програмі BLASTP амінокислотну послідовність білка, у FASTA форматі (див. файл protein ID.doc). Проведіть пошук гомологів в базі даних Swiss-Prot та заповніть перший стовпчик таблиці 3.
- Потім проведіть пошук по базах даних PDB (Protein Data Bank) та "nr" (Non-redundant protein sequences) та заповніть таблицю 3. База даних NR зберігає послідовності без повторів, що були отримані з різних БД, таких як PDB, SwissProt, PIR. Також NR враховує трансльовані послідовності з БД Genbank.

Таблиця 3- Результат пошуку за допомогою BLAST

	Пошук у БД Swiss-Prot	Пошук у БД PDB	Пошук у БД "nr"
1. Кращий результат пошуку (збігається з послідовностю заданого білка)			
Accession			
E-value			
Вага (у битах)			
Відсоток ідентичності			

2.Скільки найкращих кандидатів по гомології знайдено? (число знахідок у списку опису з E-value <$1e^{-10}$)			
3. "Гірша з задовільних" знахідка з E-value <1			
Номер знахідки у списку описів			
Accession			
E-value			
Вага (у бітах)			
Відсоток ідентичності			
Відсоток подібності			
Довжина вирівняної послідовності			
Координати вирівнювання (від-до, у запиті та в знахідці)			
Число гепів			

- Чи був знайдений вихідний білок у Swiss-Prot та "nr", а його структуру в PDB?

- Порівняйте число явних гомологів (E-value < $1e^{-10}$) при пошуку за різними БД та поясніть можливі причини відмінностей.

- Скільки всього знахідок та яке E-value останньої знахідки?

- Чим у вашому випадку була лімітована кількість результатів пошуку: значенням E-value або заданим за замовчуванням граничним розміром видачі ?

2. Пошук гіпотетичних гомологів досліджуваного білка з використанням фільтра по таксонам.

- Для досліджуваного білка B.subtilis (див. файл Homologous proteins.doc) знайти кращого гомолога в організмах таксона, філогенетично більш далекого.

Для дослідження використовуються наступні таксони:

Eukaryota (інше царство);

Actinobacteria (інший відділ того ж царства бактерій);

Clostridia (інший клас того ж відділу Firmicutes);

Lactobacillales (інший порядок того ж класу Bacilli);-

Listeriaceae (інше сімейство того же порядку Bacillales);

Geobacillus (інший рід того ж сімейства Bacillaceae);

Bacillus anthracis (інший вид того ж роду).

- Проведіть пошук гомологів заданого білка по Swiss-Prot, при цьому введіть назву таксона у віконце "Organism".

Якщо почати набирати назву таксона, то інтерфейс відразу почне підказувати варіанти; щоб погодитися з одним з них, необхідно вибрати потрібний таксон обираючи запропоновану назву. Якщо гомолог не знайдений, поверніться на сторінку запиту (гіперпосилання "blastp suite" в лівому верхньому куті), введіть назву наступного таксону. При виконанні завдання 2 необхідно заповнити таблицю 4.

Таблиця 4 - Результат пошуку в BLAST з фільтром по таксонам

Гипотетичний гомолог	
Знайдено в таксоні	
Accession	
E-value	
Вага (у бітах)	
Ідентичність (%)	
Подібність (%)	
Довжина вирівнювання	
Координати вирівнювання (від-до, у запиті і в знахідці)	
Число гепів	

- Порівняти один з результатів пошуку, що був отриманий за допомогою програми PROTEIN BLAST (вибрати одну з знахідок, що має значення $E - value < 1\, e^{-10}$), з оптимальним глобальним та локальним вирівнюваннями (можуть бути отримані за допомогою, наприклад, програми needle (глобальне вирівнювання) та water (локальне вирівнювання) пакету програм EMBOSS (http://emboss.sourceforge.net/). При цьому необхідно використовувати однакові значення штрафів за делеції та їх продовження. Тому перш за все необхідно в меню "Algorithm parameters" в рядку "Gap Costs" програми BLAST отримати інформацію щодо штрафів. Потім необхідно побудувати оптимальні локальне та глобальне вирівнювання послідовностей відповідних білків при тих же параметрах штрафів, що використовує програма Protein BLAST.

Прокоментуйте отримані вирівнювання та заповніть таблицю 5. Чому результати вирівнювання не збігаються?

Таблиця 5 – Порівняння алгоритмів вирівнювання

	BLASTP	Needle (глобальне)	Water (локальне)
Вага			
Довжина вирівнювання			
Ідентичність (%)			
Подібність (%)			
Число гепів			

Питання до самоконтролю:

1. Пошук гомологічних послідовностей білків та нуклеїнових кислот за допомогою програми Blast. Модифікації програми Blast ?
2. Статистична значущість вирівнювання. Параметри P-value, E-value, Z-score ?
3. Поняття про хешування. Функції хешування ?
4. Пояснити, чи можливо використовувати алгоритми парного вирівнювання методом динамічного програмування для задач пошуку гомологічних послідовностей в БД?

Комп’ютерний практикум №4 Ресеквенування біологічних послідовностей

Мета роботи: знайти референсний геном для бактерії E.coli O104:H4 та провести вирівнювання відповідно до референсу.

Основні задачі роботи:

1. Розглянути основні алгоритми вирівнювання, що можуть бути використані для задач ресеквенування.

2. Навчитись знаходити референсний прокаріотичний геном.

3. Навчитись проводити ресеквенування геномів прокаріот.

Теоретичні відомості

Достатньо велика кількість біологічних задач передбачає повторне визначення послідовності ДНК організму. При цьому геном організму (або дуже схожий геном) зберігається в базі даних біологічних послідовностей. Для прокаріотів це може бути секвенування геномів, що відрізняються певними властивостями (патогенністю, продуктивністю), різних штамів організмів. Для еукаріот - накопичення даних про внутрішньовидову варіабельність генома. Якщо перед дослідником стоїть завдання, що пов`язане з визначенням послідовності геному організму (схожого організму), дані про який вже є в базі даних біологічних послідовностей, її рішення істотно спрощується. У цьому випадку проводять порівняння отриманих даних з референсним геномом. Референсний геном - секвенований, зібраний та проанотований геном організму того ж виду, до якого належить аналізований зразок.

Для порівняння отриманих даних геномного секвенування з вже наявними повними геномами немає необхідності збирати геном заново (De

novo). В такій ситуації досить вирівняти (картувати) прочитання відповідно до референсної послідовності.

Для картування причитань, що були отримані методом Sangera, був розроблений набір програм, що успішно справляються з такими задачами. Серед них особливу увагу заслуговують програми: SSAHA, BLAT. Після розробки методів високопродуктивного секвенування, кількість програм для картування почало швидко зростати. Велика частина алгоритмів вирівнювання використовує додаткові структури даних - індекси. При цьому індексується або референсний геном або прочитання. В залежності від того, яку структуру даних використовує алгоритм виділяють 3 групи програм для задач ресеквенування:

1. Основані на хешуванні: ZOOM, SHRiMP, RazerS
2. Використовують суфіксні дерева: Bowtie, BWA,
4. Використовують сортування шляхом злиття: Slider

Порядок виконання роботи

Після асемблювання геному *E.coli O104:H4* ми отримали «чорновий» геном (Standard draft) у вигляді наборів скафолдів різної довжини (див. рисунок 14). Скафолд - впорядкована послідовність контигів, між якими точно встановлена відстань. Послідовність контигів, що формують скафолд точно встановлена, але невідомі послідовності нуклеотидів , що зв`язують близькі контиги між собою (позначають літерою N).

Для подальшого аналізу геному штаму *E.coli O104:H4* необхідно знайти найближчий відомий просеквенований референсній геном та впорядкувати отримані контиги патогенного штаму E.coli відповідно до нього. Один з найпростіших варіантів пошуку референсного геному – отримати найдовший з скафолдів та, за допомогою BLAST, знайти подібні

послідовності (нас цікавить повний геном (complete genome), що максимально подібний до геному *E.coli O104:H4*), в яких цей фрагмент зустрічається з великим відсотком ідентичності та мінімальним значенням E-value. Існують різні варіанти пошуку референсного геному. Для задач пошуку референсних геномів прокаріот, еукаріот та архей часто використовують фрагмент послідовності геному, що кодує 16S rRNA, яка входить до складу малої субодиниці рибосоми. Таку специфічну ділянку можна використати для визначення виду організму. При виконанні комп`ютерного практикуму спробуємо знайти референсний геном використовуючи два різні підходи.

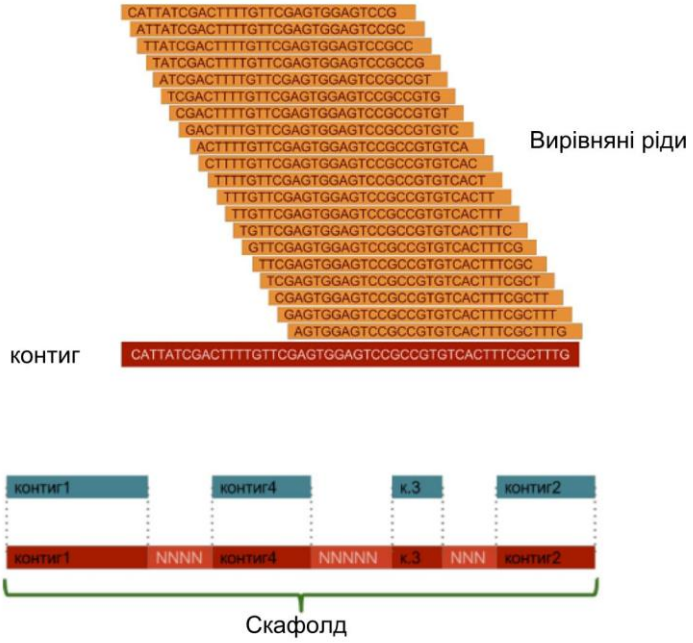


Рисунок 14 – Загальна схема асемблювання геному

Пошук референсного геному та ресеквенування

➢ Необхідно знайти найдовший скафолд з файлу scaffolds.fasta патогенного штаму E.coli та виконати пошук схожих послідовностей в БД NCBI за допомогою Nucleotide BLAST. Для більш швидкого пошуку

повних геномів в БД NCBI, необхідно в пункті Database вибрати Nucleotide collection (див рисунок 15) [4-6].

- Щоб знайти тільки ті геноми, які були присутні в базі GenBank на початок 2011 року, задайте діапазон за допомогою параметра PDAT в поле "Entrez Query": 1900/01/01: 2011/01/01 [PDAT] (див. рисунок 13)
- Виберіть найбільш близький повний геном (Ec55989), що не належить штаму E.coli O104:H4.

E.coli 5598 відноситься до ентероагрегативних E.Coli, що не повинні викликати кровотечі. Хоча геноми E.coli O104:H4 та E.coli 55989 подібні, необхідно визначити причину патогенності штаму O104:H4.

- Який у нього номер в базі GenBank (Accession Number) та номер по списку ? Яке значення E-value такої знахідки ?
- Знайдіть в мережі Інтернет інформацію про даний бактеріальний організм (див. поле description), інформацію про різновиди E.coli та патогенні штами E.coli. Які причини виникнення ешерихіозів ?
- Збережіть референсний геном у форматі FASTA. Яку довжину (пар нуклеотидів) має цей геном?

Для подальшого порівняння нам знадобиться анотація генів референсного геному.

- Збережіть файл з анатацією на комп'ютер в форматі GenBank (full). Скільки в цьому геномі кодуючих нуклеотидних послідовностей (CDS) ?
- Впорядкуємо нашу збірку по референсу за допомогою програми MEDUSA (http://combo.dbe.unifi.it/medusa). В якості вхідних даних необхідно завантажити файли чорнової збірки геному scaffolds.fasta та референсного геному. Скільки контигів знайшли своє місце на референсному геномі ?

- Збережіть упорядкований набір контигів (Scaffold fasta) для подальшого аналізу? Яку довжину має впорядкована збірка геному *E.coli O104:H4* ?

Пошук референсного геному та ресеквенування. Другий варіант пошуку.

- Знайдемо в нашому зібраному геномі фрагмент 16S рибосомальної РНК. Для пошуку цієї ділянки скористаємося сервісом RNAmmer (http://www.cbs.dtu.dk/services/RNAmmer/). Необхідно завантажити до нього файл scaffolds.fasta. Після закінчення роботи програми виберіть "Download prediction result" → "Fasta", та скопіюйте ділянку, що відповідає за кодування 16S rRNA. Який контиг містить таку ділянку?
- Для пошуку референсного геному за допомогою Nucleotide BLAST необхідно використати фрагмент геному патогенного штаму E.coli, що кодує 16S rRNA.
- Для обмеження результатів пошуку схожих геномів необхідно в пункті Database вибрати Nucleotide collection.
- Щоб знайти тільки ті геноми, які були присутні в базі GenBank на початок 2011 року, необхідно задати діапазон за допомогою параметра PDAT в поле "Entrez Query": 1900/01/01: 2011/01/01 [PDAT] (див. рисунок 15)
- Виберіть найбільш близький повний геном, що не належить штаму E.coli O104:H4.
- Перейдіть у Genbank та збережіть геном у форматі Fasta (Send - Complete Record - File – Fasta)
- Також для подальшого порівняння геномів нам знадобиться анотація генів для даного геному. Завантажте її до себе на комп'ютер в форматі GenBank (Send - Complete Record - File - GenBank (full)).
- Використовуючи програму Medusa впорядкуємо нашу збірку геному E.coli O104:H4 на референс. Програму можна завантажити на свій

комп'ютер (https://github.com/combogenomics/medusa/releases/), або використовувати онлайн-версію (http://combo.dbe.unifi.it/medusa/) [4-6].

В якості вихідних даних необхідно вказати файл нашої збірки scaffolds.fasta та завантажити референсниой геном E.coli 55989. Збережіть отримані результати для подальшого аналізу та визначення причини патогенності E.coli O104:H4.

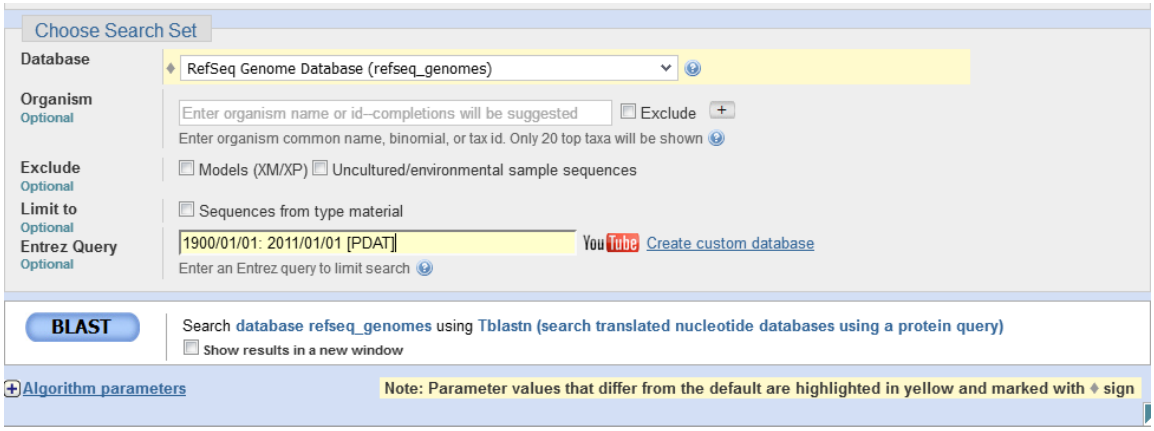


Рисунок 15 – Головна сторінка сервісу BLAST

Питання до самоконтролю:

1. Особливості картування геномів при ресеквенуванні? Основні проблеми, що виникають при повторному секвенуванні ?
2. Особливості алгоритмів, що використовують для проведення ресеквенування ?

Комп'ютерний практикум №5. Пошук кодуючих ділянок патогенного штаму e.coli.

Мета роботи: ознайомитись з основними принципами пошуку кодуючих ділянок геномів прокаріот, що кодують токсичні білки, на прикладі патогенного штаму *E.coli* O104:H4.

Основні задачі роботи:

1. Засвоїти основні принципи передбачення кодуючих ділянок (генів) ДНК на основі гомології.
2. Ознайомитись з молекулярною організацією геномів прокаріот.
3. Вміти ідентифікувати відкриті рамки зчитування (ORF).
4. Навчитися знаходити гени, що кодують токсичні білки (shigatoxin, shiga-like toxin).

Теоретичні відомості

Ген – це послідовність нуклеотидів, що кодує білковий продукт або певну функціональну РНК. Методи передбачення генів поділяються на дві групи. До першої відносять методи передбачення, що враховують молекулярну організацію геномів та властивості їх нуклеотидного складу. До другої групи відносяться методи, що базуються на подібності двох геномів (генів) – це порівняльні методи [14].

Методи передбачення генів на основі аналізу сигналів. В програмах які дозволяють ідентифікувати гени використовується велика кількість біологічно важливих сигналів, що містяться в геномних послідовностях, до яких відносять:

- *Сигнали транскрипції.* Існують три основних сигнала:

1.Точка ініціації транскрипції (transcription start site, TSS), що характеризується CAP-сигналом, що кодує перший нуклеотид першого

екзона еукаріот. Такий сигнал зазвичай представлений одиночним пуриновим нуклеотидом (A/G)

2.ТАТА-бокс, що характерний для еукаріот. Прибнов та Гилберт – бокси – для прокаріот

3. Термінація транскрипції характеризується сигналом поліаденілування, представлений гексамером AATAAA.

- *Сигнали трансляції*

1. Нуклеотидний мотив GCCACCAUGG визначає ініціацію трансляції у мРНК хребетних. Таку послідовність називають сигналом Козак. Відповідна послідовність нуклеотидів в ДНК може бути записана у вигляді мотиву GCCACCATGG

2. Термінуючий кодон TGA, TAG, TAA зустрічається зі сторони 3/ кінця кодуючої ДНК послідовності (Coding DNA sequence, CDS). При цьому необхідно враховувати той факт, що триплет TGA може кодувати амінокислоту селеноцистеїн.

- *Сигнали сплайсингу*

На етапі процесінгу (стадія сплайсингу інтронів), на рівні пре- мРНК відбувається вирізання інтронів. Цей процес каталізується комплексом– сплайсінгосомой, що здатний розпізнавати три види сайтів:

1. 5/ кінець інтрона містить динуклеотид GT (донорний сайт)
2. 3/ кінець (акцепторний сайт) характеризується наявністю динуклеотида AG та піримідин збагаченим треком (ділянка що містить нуклеотиди цитозин та урацил).
3. Точка розгалуження що ідентифікується на рівні 3/ кінця нітрону та містить нуклеотид А.

Для задач ідентифікації генів прокаріот часто використовують відкриті рамки зчитування (Open reading frame) Відкрита рамка зчитування - нуклеотидна послідовність, яка задається положенням

першого нуклеотиду в першому ініціюючому кодоні (як правило це триплет ATG) та закінчується останнім нуклеотидом термінуючого кодону (стоп триплети TAA, TAG, TGA). Не всі знайдені ORF будуть CDS, але всі CDS завжди будуть відноситись до ORF. Особливий інтерес заслуговують ORF, що мають велику довжину (100 та більше триплетів нуклеотидів). Такі протяжні ділянки геному є потенційно кодуючими та, з великою вірогідністю, будуть відноситись до CDS, що кодує ген. CDS – це не випадкові послідовності. Таки ділянки пройшли певний еволюційних шлях та підпадали дії природного добору. Тому, якщо знайдена ORF, в якій стоп триплет зустрічається приблизно через 21 кодон, буде випадковою, та її не можна вважати CDS. При визначенні ORF можливі три варіанта зчитування інформації для обох ланцюгів ДНК, тобто загальна кількість ORF дорівнює шести. Досить популярним сервісом, що дозволяє визначати ORF, є ORF Finder (https://www.ncbi.nlm.nih.gov/orffinder/). Також для задач пошуку генів через ORF використовують пакет програм EMBOSS, Ugene.

Як відомо, генетичний код є виродженим, причому для різних амінокислот ступінь надлишковості буде різною. Деякі амінокислоти кодуються тільки одним кодоном, інші - двома, трьома, чотирма або шістьма різними синонімічні кодонами. Наприклад всі триплети нуклеотидів GGA, GGC, GGG і GGT кодують одну амінокислоту гліцин Таким чином, цей набір кодонів характеризується чотирикратним виродженням. Однак, один з вироджених кодонів зазвичай вибирається (використовується) не випадково, Один з них може зустрічається значно частіше в порівнянні з іншими. Збіг частот використання кодонів може говорити про те, що даний фрагмент геному це ген. Використовуючи частоти використання кодонів, ми можемо не тільки ідентифікувати CDS,

але й визначати видову приналежність вивчаємого фрагменту геному (див. рисунок 16)

	Codon	Amino acid[2]	%[3]	Ratio[4]	Codon	Amino acid	%	Ratio	Codon	Amino acid	%	Ratio	Codon	Amino acid	%	Ratio	
U	UUU	Phe (F)	1.9	0.51	UCU	Ser (S)	1.1	0.19	UAU	Tyr (Y)	1.6	0.53	UGU	Cys (C)	0.4	0.43	U
	UUC	Phe (F)	1.8	0.49	UCC	Ser (S)	1.0	0.17	UAC	Tyr (Y)	1.4	0.47	UGC	Cys (C)	0.6	0.57	C
	UUA	Leu (L)	1.0	0.11	UCA	Ser (S)	0.7	0.12	UAA	STOP	0.2	0.62	UGA	STOP	0.1	0.30	A
	UUG	Leu (L)	1.1	0.11	UCG	Ser (S)	0.8	0.13	UAG	STOP	0.03	0.09	UGG	Trp (W)	1.4	1.00	G
C	CUU	Leu (L)	1.0	0.10	CCU	Pro (P)	0.7	0.16	CAU	His (H)	1.2	0.52	CGU	Arg (R)	2.4	0.42	U
	CUC	Leu (L)	0.9	0.10	CCC	Pro (P)	0.4	0.10	CAC	His (H)	1.1	0.48	CGC	Arg (R)	2.2	0.37	C
	CUA	Leu (L)	0.3	0.03	CCA	Pro (P)	0.8	0.20	CAA	Gln (Q)	1.3	0.31	CGA	Arg (R)	0.3	0.05	A
	CUG	Leu (L)	5.2	0.55	CCG	Pro (P)	2.4	0.55	CAG	Gln (Q)	2.9	0.69	CGG	Arg (R)	0.5	0.08	G
A	AUU	Ile (I)	2.7	0.47	ACU	Thr (T)	1.2	0.21	AAU	Asn (N)	1.6	0.39	AGU	Ser (S)	0.7	0.13	U
	AUC	Ile (I)	2.7	0.46	ACC	Thr (T)	2.4	0.43	AAC	Asn (N)	2.6	0.61	AGC	Ser (S)	1.5	0.27	C
	AUA	Ile (I)	0.4	0.07	ACA	Thr (T)	0.1	0.30	AAA	Lys (K)	3.8	0.76	AGA	Arg (R)	0.2	0.04	A
	AUG	Met (M)	2.6	1.00	ACG	Thr (T)	1.3	0.23	AAG	Lys (K)	1.2	0.24	AGG	Arg (R)	0.2	0.03	G
G	GUU	Val (V)	2.0	0.29	GCU	Ala (A)	1.8	0.19	GAU	Asp (D)	3.3	0.59	GGU	Gly (G)	2.8	0.38	U
	GUC	Val (V)	1.4	0.20	GCC	Ala (A)	2.3	0.25	GAC	Asp (D)	2.3	0.41	GGC	Gly (G)	3.0	0.40	C
	GUA	Val (V)	1.2	0.17	GCA	Ala (A)	2.1	0.22	GAA	Glu (E)	4.4	0.70	GGA	Gly (G)	0.7	0.09	A
	GUG	Val (V)	2.4	0.34	GCG	Ala (A)	3.2	0.34	GAG	Glu (E)	1.9	0.30	GGG	Gly (G)	0.9	0.13	G
	U				C				A				G				

Рисунок 16 – Таблиця розподілу кодонів білоккодуючих ділянок E.coli [15]

Порядок виконання роботи

Впорядкований набір контигів, що був отриманий при виконанні 4 комп`ютерного практикуму, необхідно описати (проанотувати). Процес анотації складається з декількох етапів. В першу чергу в нуклеотидній послідовності визначаються ділянки, які можуть відповідати генам - ORF, а також послідовності кодонів, які статистично відповідають розподілу кодонів в генах. Після того, як ці ділянки знайдені, можна визначати їх функцію шляхом пошуку гомологічних послідовностей в базі GenBank або, наприклад, окремих білкових доменів в базі Pfam.

Для анотації бактеріальних геномів зазвичай використовують сервіс RAST (**http://rast.nmpdr.org/**). Весь процес анотації виконується на сервері (для початку роботи на ньому необхідно зареєструватися). Для анотації досить завантажити впорядкований набір контигів . Для більш точної анотації потрібно вказати таксономічне положення досліджуваного організму (номер з бази NCBI Taxonomy, див. рисунок 17), інші параметри

необхідно залишити за замовчуванням. Після завершення анотації ознайомитись з результатами на сторінці (Your Jobs - Jobs Overview - View Details **)**. Завантажте на комп'ютер підсумковий файл з анотацією в форматі GenBank.

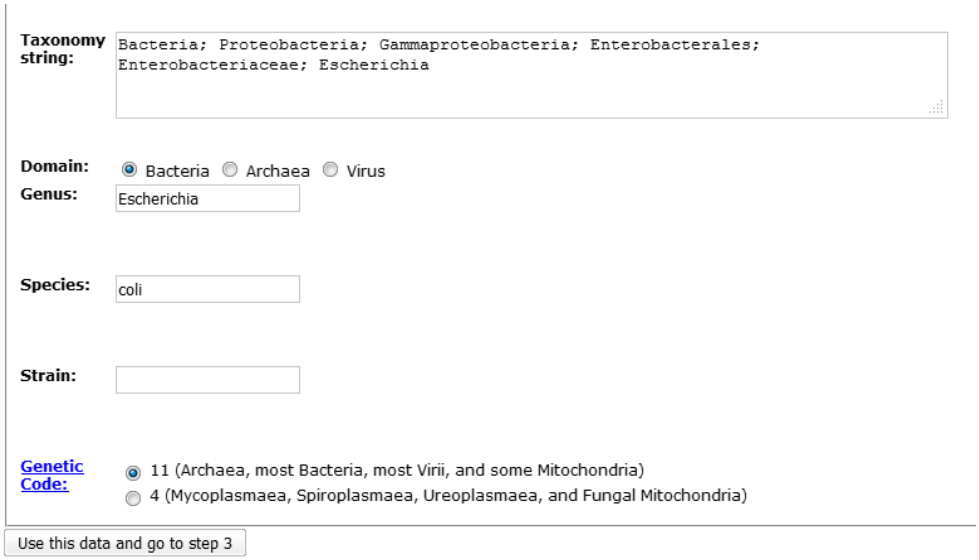


Рисунок 17 - Реєстраційна форма сервісу RAST

- За допомогою програми Mauve (http://darlinglab.org/mauve/mauve.html) порівняти два проанотованих генома (геном *E.coli* O104:H4 та референсний геном E.coli 55989). Запустити програму в режимі " Align with progressive Mauve".
- Вибрати спочатку проанотований геном патогенного штаму *E.coli* O104:H4, потім референсний геном E.coli 55989 (див. комп`ютерний практикум №4). Файли з проанотованими послідовностями E.coli 55989 та *E.coli* O104:H4 повинні бути у форматі GenBank В отриманому вирівнюванні знайти певні гени за допомогою функції Sequence Navigator (значок бінокля в верхньому рядку програми). У лівому вікні можна вибрати «Product», а в правому ввести назву гену або його функції (рисунок 18).

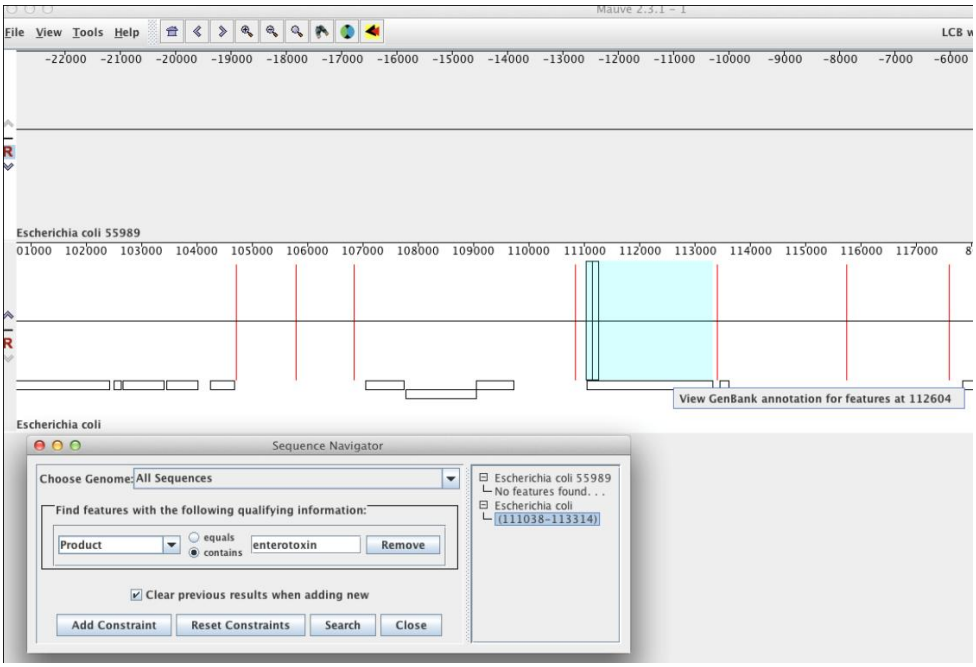


Рисунок 18 – Інтерфейс програми Mauve

Відомо, що внутрішні кровотечі при зараженні деякими штамами кишкової палички можуть бути пов`язані з певними білками – так званими шига-токсинами та шигоподібними токсинами. Бактерії можуть отримувати ці гени в результаті горизонтального переносу від інших штамів та навіть інших видів бактерій. Скільки в зібраному геномі таких генів (shigatoxin и shiga-like toxin)?

Якщо змінити масштабування (Zoom out - Ctrl + Down), можна побачити, що всі знайдені гени, що кодують токсичні продукти, знаходяться в ділянці, яка специфічна тільки для штаму E.coli O104:H4. В референсному (геномі бактерії E.coli 55989), що відносять до групи ентероагрегативних кишкових паличок (enteroaggregative E.coli, EAEC), така ділянка відсутня. Якщо подивитися на сусідні гени E.coli O104:H4, то ми побачимо, що багато з них кодують білки бактеріофагів. Відповідно до отриманих даних ми можемо зробити висновок, що проста нешкідлива кишкова паличка в певний момент часу отримала фрагмент геному від бактериофагу. Цей бактеріофаг, крім звичайних білків, містить гени

шигатоксинів. За допомогою програми Mauve знайти ділянки геному, що відповідають А та В субодиниці shiga like toxin (див. рисунок 19, 20).

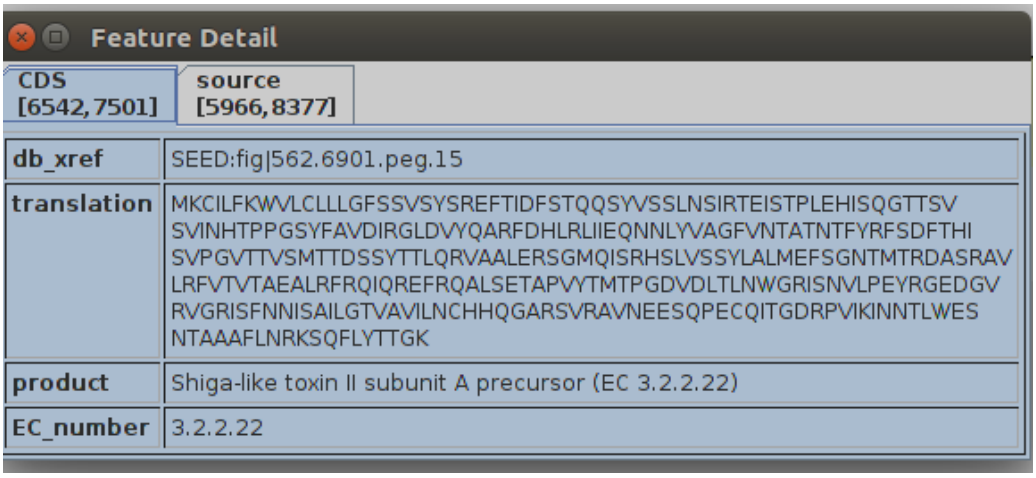


Рисунок 19 – Амінокислотна послідовність білка А субодиниці shiga like toxin

В базі даних Genbank знайти інформацію про просторову структуру білків-токсинів. Яку функцію виконує А та В субодиниця токсину патогенного штаму E.coli O104:H4. Зберегти у звіт послідовності нуклеотидів двох судобиниць shiga like toxin.

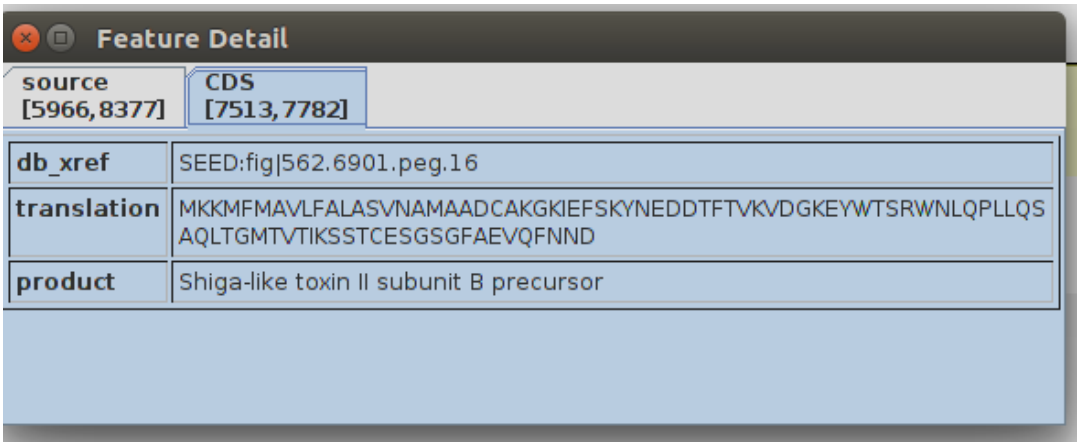


Рисунок 20 - Амінокислотна послідовність білка В субодиниці shiga like toxin

➢ За допомогою програми ResFinder перевіримо штам на стійкість до антибіотиків (https://cge.cbs.dtu.dk/services/ResFinder-2.1/). До яких антибіотиків стійкий штам E.coli O104:H4?

Питання до самоконтролю:

1. Визначення кодуючих ділянок геномів прокаріот на основі пошуку сигналів транскрипції, сплайсинга та трансляції?
2. Алгоритм пошуку генів за допомогою ORF? Особливості відкритих рамок зчитування, що відносяться до CDS?
3. Використання частот розподілу кодонів прокаріот для задач ідентифікації генів?
4. Механізми дії та класифікація сучасних антибіотиків?
5. Особливості будови та розмноження бактеріофагів?

Комп'ютерний практикум № 6. Вирівнювання амінокислотних послідовностей. Алгоритм глобального вирівнювання Нідлмана – Вунша.

Мета роботи: навчитись отримувати оптимальні глобальні вирівнювання двох послідовностей методом динамічного програмування.

Основні задачі:

1. Навчитися вирівнювати послідовності, за допомогою точкової матриці Dot Plot.
2. Ознайомитися з особливостями методу динамічного програмування
3. Вміти проводити процедуру зворотного проходу та отримувати глобальне вирівнювання гомологічних послідовностей

4. Ознайомитися з матрицями замін амінокислот PAM та Blosum та навчитись визначати вагові коефіцієнти відповідно до значень вірогідності замін амінокислот

Теоретичні відомості

Алгоритм глобального вирівнювання двох біологічних послідовностей методом динамічного програмування був вперше запропонований Солом Нідлманом та Крістіаном Вуншем (Saul B. Needlemani Christian D. Wunsch). Для ідентифікації локального збігу подібний алгоритм був вперше використаний Темплем Смітом та Міхаелем Уотерманом (Temple F. Smith і Michael S. Waterman).

Алгоритм глобального вирівнювання Нідлмана-Вунша методом динамічного програмування полягає в побудові на даному етапі оптимального вирівнювання, при цьому використовуються отримані на попередніх етапах оптимальні вирівнювання початкових фрагментів вихідних послідовностей. Для двох послідовностей x та y з елементами ($0< i < n$) та ($0 < j < m$) ми будуємо матрицю F. Елемент $F(i,j)$ цієї матриці містить вагу (рахунок, score) найкращого вирівнювання елементів $x_{1\ldots i}$ та $y_{1\ldots j}$ послідовностей x та y, відповідно. Матрицю F ми будуємо рекурсивно. Присвоїмо початковій точці нульову вагу $F\ 0{,}0\ = 0$. Далі ми заповнюємо матрицю в порядку зростання обох індексів, тобто з верхнього лівого кута до нижнього правого. Якщо вже визначені $F(i-1,j-1)$, $F(i-1,j)$, $F(i,j-1)$, то можна визначити $F(i,j)$. Можливі три варіанти отримання ваги $F(i,j)$ відповідно до трьох можливих варіантів вирівнювання (дивись рисунок 19)

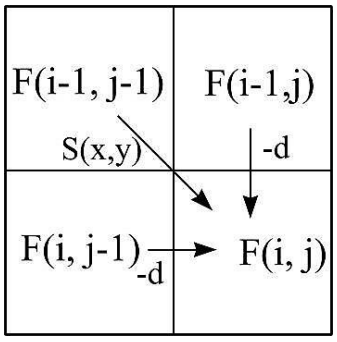


Рисунок 19 – Отримання ваги $F(i,j)$

Якщо y_i елемент однієї послідовності вирівняний з x_i елементом другої послідовності, тоді до значенням ваги $F\ i-1,j-1$ додаємо бали за вирівнювання $S(x_i+y_j)$, що можуть бути отримані з матриць PAM або BLOSUM.

$$F\ i,j\ = F\ i-1,j-1\ + S(x_i+y_j)$$

Матриці PAM (Percent Accepted Mutation) відображають вірогідність замін амінокислот в процесі еволюції (рисунок 20). Для отримання матриці PAM Маргарет Дейхофф оцінювала заміни амінокислот у групі гомологічних білків, при цьому були зафіксовані 1572 заміни у 71 групах послідовностей білків, які були подібні принаймні на 85%. Міра розбіжності послідовностей оцінюється в одиницях PAM - відсоток прийнятих (або зафіксованих) мутацій. Таким чином, дві послідовності мають відстань 1 PAM, якщо вони ідентичні на 99% (іншими словами, зафіксована одна точкова мутація на100 амінокислотних залишків).

	A	R	N	D	C	Q	E	G	H	I	L	K	M	F	P	S	T	W	Y	V
Ala (A)	**5**	−4	−2	−1	−4	−2	−1	0	−4	−2	−4	−4	−3	−6	0	1	1	−9	−5	−1
Arg (R)	−4	**8**	−3	−6	−5	0	−5	−6	0	−3	−6	2	−2	−7	−2	−1	−4	0	−7	−5
Asn (N)	−2	−3	**6**	3	−7	−1	0	−1	1	−3	−5	0	−5	−6	−3	1	0	−6	−3	−5
Asp (D)	−1	−6	3	**6**	−9	0	3	−1	−1	−5	−8	−2	−7	−10	−4	−1	−2	−10	−7	−5
Cys (C)	−4	−5	−7	−9	**9**	−9	−9	−6	−5	−4	−10	−9	−9	−8	−5	−1	−5	−11	−2	−4
Gln (Q)	−2	0	−1	0	−9	**7**	2	−4	2	−5	−3	−1	−2	−9	−1	−3	−3	−8	−8	−4
Glu (E)	−1	−5	0	3	−9	2	**6**	−2	−2	−4	−6	−2	−4	−9	−3	−2	−3	−11	−6	−4
Gly (G)	0	−6	−1	−1	−6	−4	−2	**6**	−6	−6	−7	−5	−6	−7	−3	0	−3	−10	−9	−3
His (H)	−4	0	1	−1	−5	2	−2	−6	**8**	−6	−4	−3	−6	−4	−2	−3	−4	−5	−1	−4
Ile (I)	−2	−3	−3	−5	−4	−5	−4	−6	−6	**7**	1	−4	1	0	−5	−4	−1	−9	−4	3
Leu (L)	−4	−6	−5	−8	−10	−3	−6	−7	−4	1	**6**	−5	2	−1	−5	−6	−4	−4	−4	0
Lys (K)	−4	2	0	−2	−9	−1	−2	−5	−3	−4	−5	**6**	0	−9	−4	−2	−1	−7	−7	−6
Met (M)	−3	−2	−5	−7	−9	−2	−4	−6	−6	1	2	0	**10**	−2	−5	−3	−2	−8	−7	0
Phe (F)	−6	−7	−6	−10	−8	−9	−9	−7	−4	0	−1	−9	−2	**8**	−7	−4	−6	−2	4	−5
Pro (P)	0	−2	−3	−4	−5	−1	−3	−3	−2	−5	−5	−4	−5	−7	**7**	0	−2	−9	−9	−3
Ser (S)	1	−1	1	−1	−1	−3	−2	0	−3	−4	−6	−2	−3	−4	0	**5**	2	−3	−5	−3
Thr (T)	1	−4	0	−2	−5	−3	−3	−3	−4	−1	−4	−1	−2	−6	−2	2	**6**	−8	−4	−1
Trp (W)	−9	0	−6	−10	−11	−8	−11	−10	−5	−9	−4	−7	−8	−2	−9	−3	−8	**13**	−3	−10
Tyr (Y)	−5	−7	−3	−7	−2	−8	−6	−9	−1	−4	−4	−7	−7	4	−9	−5	−4	−3	**9**	−5
Val (V)	−1	−5	−5	−5	−4	−4	−4	−3	−4	3	0	−6	0	−5	−3	−3	−1	−10	−5	**6**

Рисунок 20 – Матриця замін амінокислот PAM 70

Інше сімейство матриць замін амінокислот – матриці BLOSUM – розробили в 1992 році подружжя Стівен та Джорджа Хенікофф (StevenHenikoff, Jorja G. Henikoff) на основі бази даних BLOCKS вирівняних послідовностей білків (BLOcks SUbstitution Matrix)

	A	R	N	D	C	Q	E	G	H	I	L	K	M	F	P	S	T	W	Y	V
Ala (A)	**5**	−2	−1	−2	−1	−1	−1	0	−2	−1	−2	−1	−1	−3	−1	1	0	−3	−2	0
Arg (R)	−2	**7**	−1	−2	−4	1	0	−3	0	−4	−3	3	−2	−3	−3	−1	−1	−3	−1	−3
Asn (N)	−1	−1	**7**	2	−2	0	0	0	1	−3	−4	0	−2	−4	−2	1	0	−4	−2	−3
Asp (D)	−2	−2	2	**8**	−4	0	2	−1	−1	−4	−4	−1	−4	−5	−1	0	−1	−5	−3	−4
Cys (C)	−1	−4	−2	−4	**13**	−3	−3	−3	−3	−2	−2	−3	−2	−2	−4	−1	−1	−5	−3	−1
Gln (Q)	−1	1	0	0	−3	**7**	2	−2	1	−3	−2	2	0	−4	−1	0	−1	−1	−1	−3
Glu (E)	−1	0	0	2	−3	2	**6**	−3	0	−4	−3	1	−2	−3	−1	−1	−1	−3	−2	−3
Gly (G)	0	−3	0	−1	−3	−2	−3	**8**	−2	−4	−4	−2	−3	−4	−2	0	−2	−3	−3	−4
His (H)	−2	0	1	−1	−3	1	0	−2	**10**	−4	−3	0	−1	−1	−2	−1	−2	−3	2	−4
Ile (I)	−1	−4	−3	−4	−2	−3	−4	−4	−4	**5**	2	−3	2	0	−3	−3	−1	−3	−1	4
Leu (L)	−2	−3	−4	−4	−2	−2	−3	−4	−3	2	**5**	−3	3	1	−4	−3	−1	−2	−1	1
Lys (K)	−1	3	0	−1	−3	2	1	−2	0	−3	−3	**6**	−2	−4	−1	0	−1	−3	−2	−3
Met (M)	−1	−2	−2	−4	−2	0	−2	−3	−1	2	3	−2	**7**	0	−3	−2	−1	−1	0	1
Phe (F)	−3	−3	−4	−5	−2	−4	−3	−4	−1	0	1	−4	0	**8**	−4	−3	−2	1	4	−1
Pro (P)	−1	−3	−2	−1	−4	−1	−1	−2	−2	−3	−4	−1	−3	−4	**10**	−1	−1	−4	−3	−3
Ser (S)	1	−1	1	0	−1	0	−1	0	−1	−3	−3	0	−2	−3	−1	**5**	2	−4	−2	−2
Thr (T)	0	−1	0	−1	−1	−1	−1	−2	−2	−1	−1	−1	−1	−2	−1	2	**5**	−3	−2	0
Trp (W)	−3	−3	−4	−5	−5	−1	−3	−3	−3	−3	−2	−3	−1	1	−4	−4	−3	**15**	2	−3
Tyr (Y)	−2	−1	−2	−3	−3	−1	−2	−3	2	−1	−1	−2	0	4	−3	−2	−2	2	**8**	−1
Val (V)	0	−3	−3	−4	−1	−3	−3	−4	−4	4	1	−3	1	−1	−3	−2	0	−3	−1	**5**

Рисунок 21 – Матриця замін амінокислот BLOSUM 62

Якщо y_i елементу однієї послідовності (відповідає стовпчикам матриці) зіставлений пропуск (gap) "-" у другій послідовності x (формує рядки матриці), то за це нараховується штраф d.

$$F\ i,j\ = F\ i,j-1\ - d$$

У випадку, коли x_i елементу зіставлений пропуск в послідовності y також "нараховується" штраф.

$$F\ i,j\ = F\ i-1,j\ - d$$

Найбільшу вагу вирівнювання двох фрагментів послідовностей $x_{1\ldots i}$ (довжиною i) та $y_{1\ldots j}$ (довжиною j) визначається як максимум трьох варіантів:

$$F\ i,j\ = max \begin{matrix} F\ i-1,j-1\ + S(x_i + y_j) \\ F\ i-1,j\ - d) \\ F\ i,j-1\ - d \end{matrix}$$

Таку рекурсивну процедуру повторюємо, послідовно збільшуючи номер рядка (а всередині рядка – послідовно збільшуючи номер стовпчика), до тих пір, поки не буде заповнена вся матриця $F(i,j)$

Оскільки вздовж верхнього рядка, де $i = 0$ отримання значень $F\ i,j\ = 0$ при зміщенні зліва направо (горизонтальний перехід, див. рисунок 17) відповідає вставкам пропусків у послідовність x, встановлюємо $F\ 0,j\ = -jd$

Аналогічно вздовж лівого стовпчика де $j = 0$ отримання значень $F(i,0)$ при зміщенні зверху вниз (вертикальний перехід, див. рисунок 22) відповідає вставкам пропусків у послідовність y, встановлюємо $F\ i,0\ = -id$

Розглянемо послідовність заповнення матриці динамічного програмування на прикладі глобального вирівнювання двох послідовностей з використанням матриці замін амінокислот BLOSUM 50 (рисунок 21) та значенням величини фіксованого штрафу $d = 8$. Для

послідовностей $x = PAWHEAE$ та $y = HEAGAWGHEE$ матриця динамічного програмування буде мати вигляд (див. рисунок 22) [16].

y_j		j=0	H	E	A	G	A	W	G	H	E	E
x_i	i=0		*1*	*2*	*3*	*4*	*5*	*6*	*7*	*8*	*9*	*10*
		0	-8	-16	-24	-32	-40	-48	-56	-64	-72	-80
P	*1*	-8	-2	-9	-17	-25	-33	-41	-49	-57	-65	-73
A	*2*	-16	-10	-3	-4	-12	-20	-28	-36	-44	-52	-60
W	*3*	-24	-18	-11	-6	-7	-15	-5	-13	-21	-29	-37
H	*4*	-32	-14	-18	-13	-8	-9	-13	-7	-3	-11	-19
E	*5*	-40	-22	-8	-16	-16	-9	-12	-15	-7	3	-5
A	*6*	-48	-30	-16	-3	-11	-11	-12	-12	-15	-5	2
E	*7*	-56	-38	-24	-11	-6	-12	-14	-15	-12	-9	1

Рисунок 22 – Матриця динамічного програмування

Значення правої нижньої комірки матриці $F(n,m)$ за визначенням містить найкращу вагу вирівнювання двох послідовностей $x_{1\ldots i}$ та $y_{1\ldots j}$. Для побудови самого вирівнювання необхідно відновити послідовність виборів, яка привела від початкової точки $F\ 0{,}0\ = 0$ до фінішної точки $F(n,m)$. Процедура відновлення виборів вирівнювання називається *процедурою зворотного проходу* (traceback procedure, див. рисунок 23), при цьому ми будуємо граф зворотного проходу та, одночасно, записуємо вирівняні рядки, додаючи ліворуч до поточного вирівнювання пару символів:

$\left|\frac{y_j}{x_i}\right|$ Якщо вага була отримана з комірки $F\ i-1, j-1$

$\left|\frac{y_j}{-}\right|$ Якщо вага була отримана з комірки $F\ \ i,j-1$

$\left|\frac{-}{x_i}\right|$ якщо вага була отримана з комірки $F\ \ i-1,j$

y_j	i=0	j=0	**H**	**E**	**A**	**G**	**A**	**W**	**G**	**H**	**E**	**E**
x_i			*1*	*2*	*3*	*4*	*5*	*6*	*7*	*8*	*9*	*10*
		0	-8	-16	-24	-32	-40	-48	-56	-64	-72	-80
P	*1*	-8	-2	-9	-17	-25	-33	-41	-49	-57	-65	-73
A	*2*	-16	-10	-3	-4	-12	-20	-28	-36	-44	-52	-60
W	*3*	-24	-18	-11	-6	-7	-15	-5	-13	-21	-29	-37
H	*4*	-32	-14	-18	-13	-8	-9	-13	-7	-3	-11	-19
E	*5*	-40	-22	-8	-16	-16	-9	-12	-15	-7	3	-5
A	*6*	-48	-30	-16	-3	-11	-11	-12	-12	-15	-5	2
E	*7*	-56	-38	-24	-11	-6	-12	-14	-15	-12	-9	1

Рисунок 23 – Схема зворотного проходу

Для нашого прикладу виявляються можливими три варіанта вирівнювання з максимальною вагою, що дорівнює 1:

y	H	E	A	G	A	W	G	H	E	-	E
x	-	-	P	-	A	W	-	H	E	A	E

y	H	E	A	G	A	W	G	H	E	-	E
x	-	P	-	-	A	W	-	H	E	A	E

y	H	E	A	G	A	W	G	H	E	-	E
x	-	P	A	-	-	W	-	H	E	A	E

Порядок виконання роботи

- За допомогою матриці замін амінокислот BLOSUM 62 та фіксованого штрафу за пропуски $d = 8$, методом Нідлмана - Вунша побудувати глобальне вирівнювання двох фрагментів білків (дивись файл Table protein fragments. doc) та розрахувати рахунок (вагу) вирівнювання.
- За допомогою програми needle пакету EMBOSS отримати оптимальне глобальне вирівнювання двох послідовностей з завдання 1. При вирівнюванні використовувати матрицю BLOSUM 62 та фіксований штраф за пропуски $d = 8$. Збіглися вирівнювання чи ні ? Порівняйте вагу отриманих вирівнювань розрахованих вручну та за допомогою програми needle.

Питання для самоконтролю:

1. Які методи дають змогу кількісно оцінити відстані між двома рядками послідовностей?
2. Які існують види штрафів за делеції та вставки?
3. Яким чином в матриці dot plot розрізняють делеціїї та вставки?
4. Характеристика основних напрямів руху в матриці вирівнювання?
4. Алгоритм заповнення комірок матриці $F(i, j)$ динамічного програмування при застосуванні алгоритму глобального вирівнювання?
5. У чому полягає суть процедури зворотного проходу?
6. Чому виникають різні варіанти вирівнювань, що мають однакову вагу?

Комп'ютерний практикум № 7. Вирівнювання амінокислотних послідовностей. Алгоритм локального вирівнювання Сміта - Уотермана.

Мета роботи: навчитись знаходити оптимальне локальне вирівнювання двох послідовностей методом динамічного програмування.

Основні задачі:

1. Ознайомитися з алгоритмом заповнення матриці динамічного програмування відповідно до локального вирівнювання
2. Вміти проводити процедуру зворотного проходу та отримувати оптимальне локальне вирівнювання біологічних послідовностей

Теоретичні відомості

Алгоритм Сміта - Уотермана використовують, коли необхідно знайти оптимальне вирівнювання підпослідовностей (subsequences) вихідних послідовностей x та y, наприклад коли існує необхідність знайти загальні домени гомологічних білків. Локальне вирівнювання також є ефективним способом виявлення подібності при порівнянні послідовностей, що зазнали суттєвих змін (дивергенція ознак) під дією рушійних сил еволюції. Найкращим локальним вирівнюванням називається вирівнювання підпослідовностей x та y, що має найбільшу вагу. У цілому алгоритм локального вирівнювання подібний алгоритму глобального вирівнювання, але в ньому існують відмінності.

Для отримання значення ваги для кожного елемента матриці динамічного програмування добавлена можливість присвоєння нульового ваги $F(i,j) = 0)$ у разі, якщо всі інші варіанти вибору дають негативні значення. Якщо найкраще вирівнювання на даному етапі має негативну вагу, то починають нове вирівнювання. Всякий раз, коли найбільша вага

комірки матриці виявляється негативною, в комірку записується значення 0 та починається нове вирівнювання. Внаслідок додавання нового «нульового» вибору, елементи верхнього рядку та лівого стовпця в даному алгоритмі також прирівнюються нулю, а не $-id$ та $-jd$, як при глобальному вирівнюванні. В результаті будь-яка підпослідовність може "ковзати" вздовж іншої перед початком вирівнювання без додавання штрафів за вставки пропуски (делеції).

Локальне вирівнювання може закінчиться в будь-якому місці матриці, а не тільки в правому нижньому кутку, як у випадку глобального вирівнювання. Таким чином, замість $F(n,m)$ найкращою вагою при локальному вирівнюванні може бути найбільше значення $F\ i,j$ матриці та процедуру зворотного проходу потрібно починати саме з цього місця. Процедуру зворотного проходу закінчуємо, як тільки зустрічається нульовий елемент матриці, що відповідає початку вирівнювання (див. рисунок 24).

y_j	i=0	j=0	H	E	A	G	A	W	G	H	E	E
x_i			*1*	*2*	*3*	*4*	*5*	*6*	*7*	*8*	*9*	*10*
		0	0	0	0	0	0	0	0	0	0	0
P	*1*	0	0	0	0	0	0	0	0	0	0	0
A	*2*	0	0	0	5	0	5	0	0	0	0	0
W	*3*	0	0	0	0	2	0	20	12	0	0	0
H	*4*	0	10	2	0	0	0	12	18	22	14	6
E	*5*	0	2	16	8	0	0	4	0	18	28	20
A	*6*	0	0	8	21	13	5	0	4	10	20	27
E	*7*	0	0	6	13	18	12	4	0	4	16	26

Рисунок 24 – Локальне вирівнювання методом Сміта – Уотермана. Схема зворотного проходу

Результатом локального вирівнювання двох послідовностей x та y є підмножина глобального вирівнювання двох послідовностей $x =$ *PAWHEAE* та $y =$ *HEAGAWGHEE* (дивись комп`ютерний практикум №6)

y	A	W	G	H	E
x	A	W	-	H	E

Порядок виконання роботи

- Використовуючи матрицю замін амінокислот BLOSUM 62 та фіксований штраф за пропуски $d = 8$, методом Сміта - Уотермана побудувати локальне вирівнювання двох фрагментів білків (*дивись файл Table protein fragments. doc*) та розрахувати рахунок (вагу) вирівнювання.

- Використовуючи матрицю замін амінокислот BLOSUM 62 та фіксований штраф за пропуски $d = 8$ за допомогою програми water пакету EMBOSS побудуйте локальне вирівнювання двох послідовностей гомологічних білків двох фрагментів білків (*дивись файл Table protein fragments. doc*). Результати вирівнювання зберегти у форматах markx0, markx1, markx2, srspair та score. Порівняти результати, що були збережені у різних форматах.
- Побудуйте локальне вирівнювання послідовностей гомологічних білків за допомогою програми matcher пакету EMBOSS та отримайте три найкращі варіанти вирівнювання з найбільшою вагою.
- Порівняйте вирівняні послідовності, що були отримані за допомогою програм needle (комп`ютерний практикум №6) та water.

Питання для самоконтролю:

1. Чим відрізняються методи вирівнювання Нідлмана-Вунша та Сміта-Уотермана ?

2. Алгоритм отримання ваги $F(i,j)$ при локальному вирівнюванні ?
3. Алгоритм отримання вагових коефіцієнтів PAM відповідно до значень вірогідності замін амінокислот ?
4. В яких випадках краще використовувати матриці PAM, а в яких Blosum ?

Комп'ютерний практикум № 8. Модифіковані алгоритми парного вирівнювання

Мета роботи: провести аналіз алгоритмів вирівнювання біологічних послідовностей з використанням модифікованих алгоритмів локального та глобального вирівнювання.

Основні задачі:

1. Ознайомитися з алгоритмом заповнення матриці динамічного програмування відповідно до модифікованих алгоритмів локального та глобального вирівнювання
2. Ознайомитись з особливостями ініціалізації матриці динамічного програмування для модифікованого алгоритму локального та глобального вирівнювання
3. Вміти проводити процедуру зворотного проходу та отримувати оптимальні вирівнювання

Теоретичні відомості

Якщо біологічні послідовності, для яких необхідно отримати оптимальне локальне вирівняння, мають велику довжину, можливо, існує велика кількість локальних вирівнювань, що мають значну вагу. В такій ситуацій для об`єктивної оцінки результатів вирівнювання, необхідно

враховувати такі локальні збіги. Прикладом може бути білок, що містить велику кількість копій одного домену або мотиву. Модифікований алгоритм локального вирівнювання дозволяє знаходити такі повтори. Цей метод знаходить одну або більше копій фрагментів однієї послідовності в інший.

Нехай y – це послідовність, що містить домен або мотив, а x - послідовність, в якій ми шукаємо копії ділянок $\boldsymbol{y}$**.** У фінальному вирівнюванні послідовність x буде розділена на ділянки, що вирівняні з фрагментами послідовності y. Між вирівняними фрагментами послідовності x можуть бути присутні розриви (не вирівняні ділянки). Схематично такі ділянки позначають крапкою. Під вагою закінченої ділянки вирівнювання ми будемо розуміти її вагу, що може бути обчислена за допомогою базового алгоритму (локального вирівнювання) мінус граничне значення T.

Модифікований алгоритм локального вирівнювання:

1. $F\ 0{,}0\ = 0$

2. $F\ i, 0\ = 0$

3. Перша комірка кожного стовпчика (перший рядок) заповнюється відповідно до співвідношення:

$$F\ 0, j\ = max \begin{matrix} F\ 0, j-1 \\ F\ i, j-1\ - T \end{matrix} \qquad (8.1)$$

4. Вагу комірок, що залишилися, отримують з співвідношення:

$$F\ i, j\ = max \begin{matrix} F(0, j) \\ F\ i-1, j-1\ + s(x_i, y_j) \\ F\ i-1, j\ - d \\ F\ i, j-1\ - d \end{matrix}$$

5. Матриця динамічного програмування заповнюється зверху вниз

Повну вагу вирівняних ділянок отримують через додавання додаткової комірки $F(0, m+1)$, вагу якої визначають відповідно до рівняння 8.1. Окремі оптимальні локальні вирівнювання можуть бути отримані процедурою зворотного проходу від комірки $F(0, m+1)$ до $F(0,0)$ [12].

Заповнимо матрицю динамічного програмування з повторами для двох тестових послідовностей $x - HEAGAWGHEE$ та $y - PAWHEAE$. При $\mathrm{T} = 20$, матриця замін амінокислот $Blosum\ 50$ (рисунок 22).

		H	E	A	G	A	W	G	H	E	E
	0	0	0	0	1	1	1	1	1	3	9
P	0	0	0	0	1	1	1	1	1	3	9
A	0	0	0	5	1	6	1	1	1	3	9
W	0	0	0	0	2	1	21	13	5	3	9
H	0	10	2	0	1	1	13	19	23	15	9
E	0	2	16	8	1	1	5	11	19	29	21
A	0	0	8	21	13	6	1	5	11	21	28
E	0	0	6	13	18	12	4	1	5	17	27

Рисунок 22 – Матриця динамічного програмування

Після проведення процедури зворотного проходу отримаємо оптимальне вирівнювання :

y	H	E	A	G	A	W	G	H	E	E
x	H	E	A	.	A	W	-	H	E	.

Ще один варіант пошуку застосовується в ситуації, коли одна послідовність містить іншу або вони перекриваються. Такий варіант часто зустрічається при порівнянні схожих фрагментів геномної ДНК або при

порівнянні фрагмента з послідовністю ДНК хромосоми. При цьому можуть виникати декілька варіантів вирівнювання (див. рисунок 23).

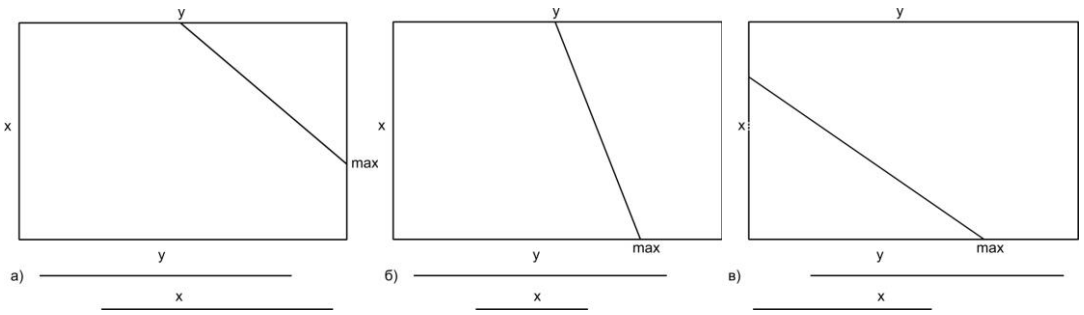


Рисунок 23 – Варіанти вирівнювання

Необхідно отримати модифікований алгоритм глобального вирівнювання, який не штрафував би фрагменти послідовностей, що не перекриваються.

Модифікований алгоритм глобального вирівнювання:

1. $F\ i,0\ = 0$
2. $F\ 0,j\ = 0$
3. $F\ i,j\ = max \begin{matrix} F\ i-1,j-1\ + S(x_i + y_j) \\ F\ i-1,j\ - d) \\ F\ i,j-1\ - d \end{matrix}$

Процедуру зворотного проходу необхідно виконувати від максимального значення відповідної комірки до верхньої горизонтальної або лівої вертикальної межі матриці динамічного програмування.

Порядок виконання роботи

- За допомогою модифікованого алгоритму локального вирівнювання (вирівнювання з повторами) визначити фрагмент (паттерн) послідовності $\boldsymbol{x}$ (дивись файл SEQ.docx, таблиця 1), що зустрічається один або декілька разів у послідовності $\boldsymbol{y}$. При вирівнювання використовувати матрицю замін амінокислот Blosum 62 при T=20. Чи зміниться результат

вирівнювання, якщо збільшити порогове значення T ? При якому значенні T було знайдена максимально кількість повторів?

- Порівняйте результати вирівнювання, що були отримані при використанні двох алгоритмів: 1.Модифікованого локального вирівнювання, що враховує повтори 2. Базового алгоритму локального вирівнювання.

Питання для самоконтролю:

1. Особливості ініціалізації матриці динамічного програмування для модифікованого алгоритму локального вирівнювання ?
2. Особливості ініціалізації матриці динамічного програмування для модифікованого алгоритму глобального вирівнювання ?
3. Процедура зворотного проходу для модифікованих алгоритмів глобального та локального вирівнювань ?

Комп'ютерний практикум № 9. Вирівнювання біологічних послідовностей з використанням афінної штрафної функції.

Мета роботи: вирівняти біологічні послідовності з використанням афінної штрафної функції

Основні задачі:

1. Ознайомитися з алгоритмом заповнення матриці динамічного програмування для алгоритму вирівнювання з афінною штрафною функцією.
2. Ознайомитись з особливостями ініціалізації матриці динамічного програмування
3. Навчитись отримувати оптимальні вирівнювання відповідно до зворотного проходу.

Теоретичні відомості

Сучасні досягнення еволюційної біоінформатики та філогенетичного аналізу накладають деякі обмеження в використання фіксованого штрафу для отримання оптимального вирівнювання біологічних послідовностей. Такі особливості пов`язані з тим, що, наприклад, при еволюції білків гомологів, зазвичай, відбуваються множинні мутації, результатом яких є видалення невеликого фрагменту амінокислотної або нуклеотидної послідовності. Для отримання оптимального вирівнювання необхідно використовувати афінну штрафну функцію, що максимально штрафує відкриття пропуску та мінімально його продовження на кожний наступний gap (розрив). Афінна штрафна функція має вигляд $\gamma\ g\ = -d - (g-1)e$, де d – штраф за відкриття пропуску, e −штраф за подовження пропуску, g −кількість пропусків.

Для такого вигляду штрафної функції час роботи алгоритму динамічного програмування залишається рівним $O\ n^2$.Однак тепер для кожної комірки F i, j матриці повинні бути збережені три значення замість одного (алгоритм локального та глобального вірування та модифіковані алгоритми). Процес вибору для трьох основних величин, відповідає трьом різним ситуаціям (таблиця 6)

Таблиця 6 – Варіанти вирівнювання

I	G	A	x_i		A	I	G	A	x_i		G	A	x_i	−	−
L	G	V	y_j		A	V	y_j	−	−		S	L	G	V	y_j

Позначимо через $M(i,j)$ найбільшу вагу за умови, що символ x_i вирівняний з y_j (таблиця 6, випадок зліва). $I_x(i,j)$- найбільша вага за

умови, що символу x_i відповідає делеція. $I_y(i,j)$- найбільша вага за умови, що символу y_j відповідає лелеці (таблиця 6, випадок справа). Розглянемо алгоритм глобального вирівнювання з використанням афінної штрафної функції.

Ініціалізація матриці:

$M\ 0{,}0\ = 0\ I_x\ 0{,}0\ = I_y\ 0{,}0\ = -\infty$

$I_x\ i,0\ = -d\ i-1\ e, M\ i,0\ = I_y\ i,0\ = -\infty$ (перший стовпчик)

$I_y\ 0,j\ = -d\ i-1\ e, M\ 0,j\ = I_x\ 0,j\ = -\infty$ (перший рядок)

Рекурсивні співвідношення:

$$M\ i,j\ = max\begin{matrix} M(i-1,j-1)+s(x_i,y_j) \\ I_x\ i-1,j-1\ +s\ x_i,y_j \\ I_y\ i-1,j-1\ +s\ x_i,y_j \end{matrix}$$

$$I_x\ i,j\ = max\begin{matrix} M\ i-1,j\ -d \\ I_x\ i-1,j\ -e \end{matrix}$$

$$I_y\ i,j\ = max\begin{matrix} M\ i,j-1\ -d \\ I_y\ i,j-1\ -e \end{matrix}$$

Процедуру зворотного проходу виконуємо від максимального значення $M\ m,n\ , I_x\ m,n\ , I_y\ m,n\ $, закінчуємо в $M\ 0{,}0\ , I_x\ 0{,}0\ , I_y\ 0{,}0$

Розглянемо алгоритм локального вирівнювання з використанням афінної штрафної функції.

Ініціалізація матриці:

$M\ 0{,}0\ = M\ i,0\ = M\ 0,j\ = 0$

$I_x\ 0,j\ = I_x\ i,0\ = I_y\ 0,j\ = I_y\ i,0\ = -\infty$

Рекурсивні співвідношення ідентичні алгоритму глобального вирівнювання.

Процедуру зворотного проходу виконуємо від максимального значення $M\ i,j$ та закінчуємо в $M\ i,j$.

Систему рекурсивних співвідношень можна представити за допомогою схеми (рисунок 24). На схемі показані стани для трьох значень матриці зі стрілками переходів з одного стану в інший. У теорії алгоритмів така система називається скінченним автоматом. Вирівнювання відповідає шляху через стани автомату [16].

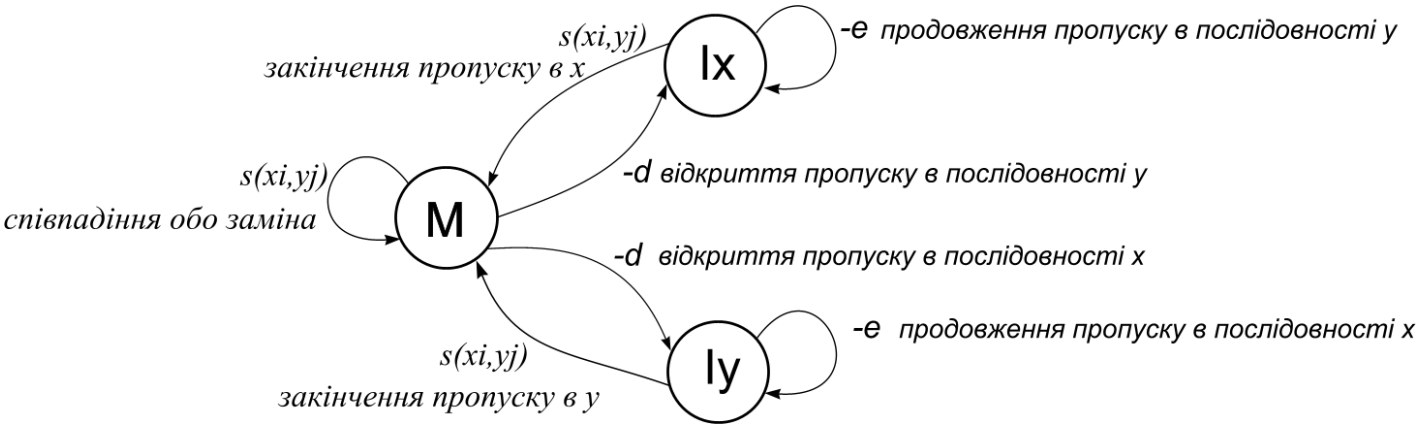


Рисунок 24 – Діаграма взаємозв`язків між станами

Порядок виконання роботи

- Розрахувати вагу оптимального глобального та локального вирівнювання двох заданих гомологічних послідовностей використовуючи матрицю BLOSUM62 та афінний штраф за пропуски $\gamma\ g\ = -d - (g - 1)e$
- Порівняйте отримані результати з оптимальним глобальним та глобальним вирівнюванням (фіксований штраф за пропуски).

Питання для самоконтролю:

1. Особливості ініціалізації матриці динамічного програмування для алгоритму вирівнювання з афінною функцією штрафів ?
2. Рекурсивні співвідношення алгоритму вирівнювання з афінною штрафною функцією?
3. Процедура зворотного проходу алгоритму глобального та локального вирівнювання з афінною штрафною функцією?

Комп'ютерний практикум № 10. Банк даних білкових послідовностей UniProt

Мета роботи: ознайомитись з особливостями збереження молекулярно-біологічної інформації в банку даних Uniprot та вміти знаходити інформацію про досліджуваний білок.

Основні задачі:

1. Ознайомитися з особливостями збереження молекулярно-біологічної інформації в БД Uniprot.
2. Ознайомитися з структурою ідентифікатора БД Uniprot та особливостями запису інформації у Fasta форматі.
3. За допомогою відповідних міток полей БД Uniprot вміти знаходити необхідну інформацію про білок.
4. Навчитись визначати конформацію білкової молекули.

Теоретичні відомості

Як і більшість архівних БД біомедичного спрямування (БД DDBJ, БД GenBank, БД EMBL) є двовимірними (тобто одному полю відповідає один запис), а їх функціональність розширюється за рахунок

використання різних ідентифікаторів (в межах одного запису). Однак на відміну від архівних БД кожен запис оцінюється експертом, який визначає ступінь достовірності як функцій самої органічної сполуки (білка для БД UniProt), так і функціональних властивостей окремих його частин. Банк даних UniProt об`єднує два банка білкових послідовностей: 1. Swissprot – БД, що керується та містить найбільш достовірну інформацію про білки. 2 TrEMBL – архівна БД, що містить автоматично проанотовані не перевірені записи. У таблиці 7 наведений опис рядків запису білкової БД UniProt.

Таблиця 7 – Ключові слова та опис відповідних полів БД UniProt

Code (мітка)	Content	Description	Опис
ID	Identification	Contains identifying information and characteristics of the sequence	Ідентифікатор біологічної послідовності, часто кодує біологічно осмислену інформацію, може змінюватись від випуску до випуску БД.
AC	Accession number(s)	Release-to-release stable identifiers	"Код доступу" - унікальний ідентифікатор послідовності, не змінюваний від випуску до випуску БД
DT	Date	When the entry was created, or when the sequence or annotation was modified	Дата створення запису і останньої модифікації
DE	Description	The name of the protein, often a function indicator	Назва (короткий опис) білка, часто вказує на його функцію

GN	Gene name(s)	The gene(s) that code for the protein	Ім'я гена (-ів), що кодує білок
OS	Organism species	The organism from which the sequence is derived	Організм, в якому знайдений білок
OG	Organelle	If the sequence is non-chromosomal in origin	Якщо ген білка не знаходиться в хромосомі
OC	Organism classification	The taxonomic class to which the organism belongs	Повна таксономія організму
OX	Taxonomy cross-reference(s)	The NCBI TaxID for the OC line	Ідентифікатор таксона
RN	Reference number	The sequential number of the literature citation within the entry	Номер посилання на наукову публікацію
RP	Reference position	The type of data, and the position in the sequence to which the citation refers	Тип інформації в даному посиланні та фрагмент послідовності, до якої вона відноситься.
RC	Reference comment(s)	Comments relevant to the reference cited	Коментарі до роботи, що цитується
RX	Reference cross-reference(s)	Bibliographic cross-reference, such as PubMed ID	Бібліографічні перехресні посилання, наприклад, ідентифікатор статті в PubMed
RA	Reference authors	Authors of the citation	Автори публикації
RT	Reference title	Title of the citation	Назва статті

RL	Reference location	Source of the citation, such as journal, book, or unpublished data	Джерело цитування: журнал, книга чи неопубліковані дані
CC	Comments or notes	Free text notes about the protein	Комментарі по даному білку
DR	Database cross-references	Pointers to sources or related information for the entry	Гіперпосилання на відповідні записи інших БД
PE	Protein existence	Gives indication on the evidences that we currently have for the existence of a protein	Яким чином отримана інформація про існування даного білка
KW	Keywords	Indexable indicator of function, structure, or other information	Ключові слова (індексований укажчик функції, структури та іншої інформації по білок)
FT	Feature table data	Annotation of specific residues of the sequence	Опис особливих позицій чи властивостей послідовності, що вказує на тип властивості.
SQ	Sequence header	Marks the beginning of the sequence and provides summary data	Метка начала последовательности и некоторых ее общих свойств (в частности, молекулярной массы)
(blanks)	Sequence data	The sequence itself	Амінокислотна послідовність
//	Termination line		Мітка кінця запису

Порядок виконання роботи

- Отримайте з БД UniProt (http://www.uniprot.org/) документ, що містить інформацію про білок. Для цього необхідно дописати ".txt" в адресному рядку браузера після відповідного номера послідовності.
- Уважно вивчіть отриманий документ та заповніть таблицю 8.

Таблиця 8

	Мітка поля	Зміст
Код(и) доступу ("Accession number")		
Идентифікатор запису в БД		
Назва (короткий опис) білка		
Назва гену		
Дата створення документа		
Дата останнього виправлення анотації		
Число публікацій, що були використані при створенні документа		
Журнали та рік самої пізньої публікації		
Ключові слова		
Що містить поле коментарів ?		
Ідентифікатор запису БД PDB		

- Користуючись пошуком на сайті UniProt визначити, скільки записів в UniProt описує білки приблизно з тим же описом (поле DE), що і заданий білок та заповнити таблицю 9.

Таблиця 9

Запит	Кількість записів Swissprot	Кількість записів TrEMBL

- Перегляньте список записів UniProt, що містять інформацію про ті білки, чий опис подібний до опису вашого білка (див. завдання 3). Виберіть один білок (по можливості з найбільш схожим описом, але з іншого організму та з БД TREMBL) та отримайте повний текст відповідного запису. Заповніть таблицю 10. У ліву колонку (білок 1) внести відомості про білок (див. файл Student protein), а в праву (білок 2) - про обраний білок, інформація про який зберігається в БД Trembl.

Таблиця 10

	Мітка поля	Білок 1	Білок 2
Код доступу			
Ідентифікатор послідовності в БД			
Назва (короткий опис) білка			
Дата створення документа			
Дата останнього виправлення анотації			
Назва організму			
Класифікація організму (список таксонів)			
Довжина послідовності			
Молекулярна маса білка			

Число публікацій, що були використані при створенні			
Опис вторинної структури			
Коментарі по даному білку			
Ідентифікатор запису БД PDB			

Питання до самоконтролю:

1. Структура ідентифікатора БД Uniprot ?
2. Характеристика вторинної структури білкової молекули ?
3. Особливості надвторинного та доменного рівня організації білкової молекули ?
4. Роль слабких взаємодій в стабілізації просторової структури білкової макромолекул ?
5. Класифікація сучасних баз даних біологічних послідовностей ?

Комп'ютерний практикум № 11. Визначення редакційної відстані між біологічними послідовностями. Алгоритм Вагнера–Фішера. Вирівнювання біологічних послідовностей з лінійною пам`яттю. Алгоритм Міллера-Майєрса

Мета роботи: навчитись вирівнювати біологічні послідовності з лінійною пам`яттю та визначати відстань Левенштайна.

Основні задачі:

1. Ознайомитись з методами оцінки складності алгоритмів.
2. Навчитись визначати редакційну відстань відповідно до заповненої матриці динамічного програмування.
3. Вирівнювати біологічні послідовності за лінійну пам`ять.

4. Реалізувати алгоритм оптимізації по пам`яті для глобального вирівнювання з фіксованою штрафною функцією.

Теоретичні відомості

Редакційна відстань або відстань Левенштайна – метрика що дозволяє визначити на скільки подібні дві послідовності. Редакційна відстань – мінімальна кількість операцій редагування (вставка, видалення або заміна символа), що дозволяє перетворити одну послідовність в іншу.

Алгоритм *Вагнера – Фішера* методом динамічного програмування дає можливість визначити редакційну відстань та дозволяє отримати редакційний припис (послідовність операцій редагування).

Ініціалізація матриці:

$$D\ 0,0\ = 0$$
$$D\ i,0\ = i,$$
$$D\ 0,j\ = j$$

Будь-яку послідовність довжиною i або j можливо отримати з пустого рядку за рахунок вставки необхідної кількості символів [17].

Рекурсивні співвідношення.

$D\ i,j\ = D\ i-1,j-1\$, якщо символи збігаються

Якщо символи відрізняються один від одного матрицю динамічного програмування заповнюємо рекурсивно відповідно до співвідношення:

$$D\ i,j\ = min \begin{matrix} D\ i-1,j\ +1 \\ D\ i,j-1\ +1 \\ D\ i-1,j-1\ +1 \end{matrix}$$

Вага кожної комірки на даному етапі заповнення матриці відображає кількість операцій редагування. Алгоритм дозволяю знайти мінімальну кількість операцій редагування та враховує: а) видалення символу $D\ i-$

$1, j$ (вертикальний перехід), б) додавання символу $D\ i, j-1$ (горизонтальний перехід) або в) заміна символів $D\ i-1, j-1$ (діагональний перехід в матриці динамічного програмування)

Для послідовностей ATGC та ATA знайдемо редакційну відстань та редакційний припис.

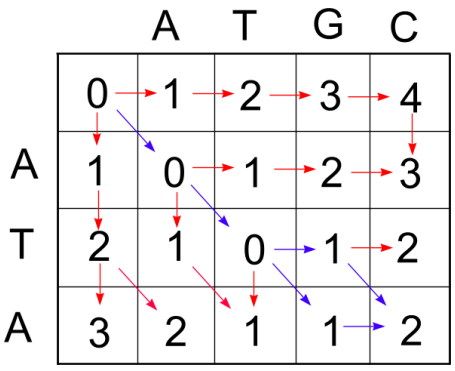


Рисунок 25 – Матриця динамічного програмування для визначення редакційної відстані

Значення ваги правої нижньої комірки матриці дозволяє визначити редакційну відстань, що для даного прикладу відповідає двом операціям редагування (див. рисунок 25). Процедура зворотного проходу дозволяє визначити два редакційних приписи (позначено переходами синього кольору). При виконанні процедури зворотного проходу видалення символу в другій послідовності (послідовність, що формує рядки матриці (ATA) пов`язана з вертикальним переходом). Горизонтальний перехід пов`язаний з вставкою символу у послідовність 2.

Перший варіант редакційного припису пов`язаний с заміною A на G та вставкою C у послідовність 2 (ATA)

A	T	G	C
A	T	A	C

Другий – з заміною A на C та вставкою G в послідовність ATA

A	T	G	C
A	T	G	A

Алгоритми парного вирівнювання методом динамічного програмування (локальне, глобальне, модифіковані алгоритми та інше) «переглядають» всі вершини графа вирівнювання. В кожній вершині графа (відповідна комірка матриці динамічного програмування) необхідно виконати три порівняння. При цьому час роботи алгоритму буде квадратичним по часу роботи $T = O(n \cdot m)$.

Іншим обчислювальним ресурсом, що може обмежити використання методів динамічного програмування для вирівнювання послідовностей є пам'ять. Для запам'ятовування ваги та відновлення оптимального вирівнювання необхідно для кожної вершини запам'ятовувати її вагу та напрямок переходу. Таким чином алгоритм пошуку редакційної відстані квадратичний по пам'яті. Враховуючи той факт, що біологічні послідовності мають велику довжину (наприклад геном людини записаний 3млрд. нуклеотидів) для таких алгоритмів необхідна оптимізація.

Розглянемо алгоритм Міллера-Майєрса на прикладі визначення оптимального глобального вирівнювання двох фрагментів білкових послідовностей (Blosum 62, $d = 8$) WHEA та AHEGP. При заповненні матриці динамічного програмування ми будемо використовувати метод оптимізації, що базується на принципі «розділяй та володарюй». При заповненні першої половини матриці динамічного програмування відсутня необхідність запам`ятовування ваги комірок попереднього стовпчика, рухаючись зліва направо зверху вниз. По мірі обчислення значень $F(i,j)$, значення ваги комірок $F(i,j-1)$ можна видаляти. Аналогічно

заповнюємо праву частину основної матриці динамічного програмування (див. рисунок 26).

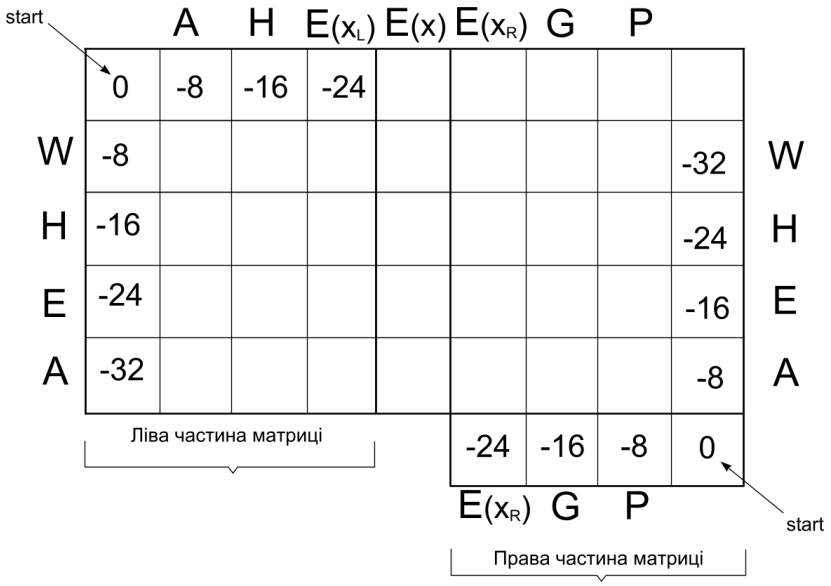


Рисунок 26- Ініціалізація матриці динамічного програмування

Розраховуємо вагу комірок рухаючись з правої нижньої частини матриці у напрямку лінії поділу. Для кожної точки x лінії поділу знаходимо вагу оптимальних вирівнювань заповнюючи матрицю з лівого верхнього кута матриці до лінії поділу x_L, кількість елементів якої дорівнює n (відповідає кількості символів послідовності x). Другу частину матриці заповнюємо до тих пір, поки не буде визначена вага комірок на лінії поділу x_R. Вага вирівнювання, що проходить через точку x дорівнює $F(x) = x_L + x_R$. Вага комірок, через яку проходить зворотний шлях для оптимального вирівнювання $F_{op} = \max(F\ x)$. Знайдена точка x поділяє матрицю на чотири квадранта, два з яких не містять комірки, через які буде проходити зворотний шлях (див. рисунок 27). На рисунку 27 червоним кольором позначені квадранти, що містять комірки, які

відповідають процедурі зворотного проходу. Комірки матриці, що позначені сірим кольором неінформативні.

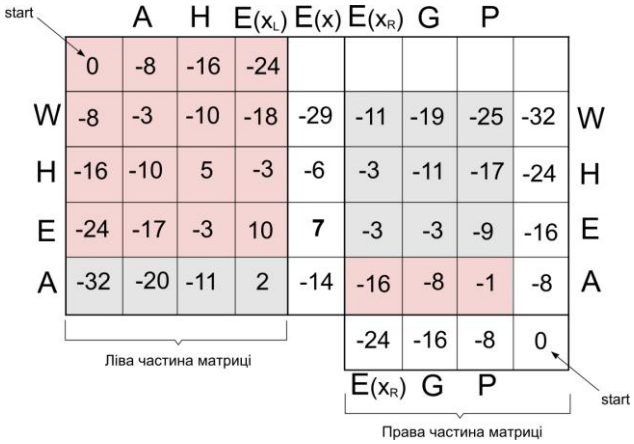


Рисунок 27 –Матриця динамічного програмування після першого поділу послідовності y

Продовжуючи процедуру поділу квадрантів навпіл знайдемо комірки, через які буде проходити зворотний шлях (див. рисунок 28, 29).

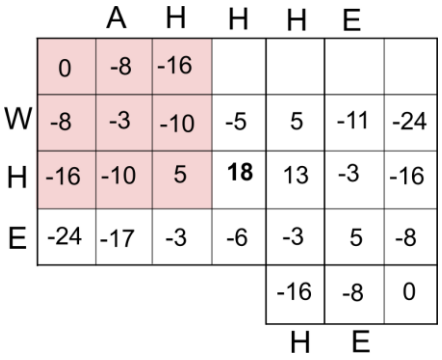


Рисунок 28 – Матриця динамічного програмування (другий поділ послідовності y)

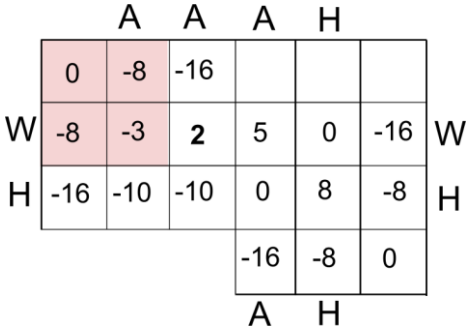


Рисунок 29 – Матриця динамічного програмування (третій поділ послідовності y)

Після виконання послідовних операцій поділу матриці та відповідних розрахунків, отримаємо комірки, рухаючись через які, при виконанні процедури зворотного проходу, отримаємо оптимальне вирівнювання двох послідовностей (рисунок 30).

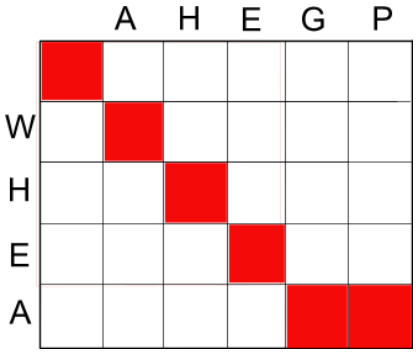


Рисунок 30 – Матриця динамічного програмування з координатами оптимального вирівнювання

Порядок виконання роботи

- За допомогою ідентифікатора Uniprot зберегти інформацію про білок у форматі FASTA. За допомогою алгоритму BLAST знайти найближчого гомолога для відповідного білка але з іншого організму.
- Розрахувати відстань Левенштайна та редакційний припис для двох фрагментів білкових послідовностей.
- Використовуючи алгоритм оптимізації Міллера-Майєрса вирівняти фрагменти послідовностей білків. Порівняйте результат отриманого вирівнювання з результатом пошуку BLAST.
- Реалізувати алгоритм оптимізації пам`яті для глобального вирівнювання з фіксованою штрафною функцією.

Питання до самоконтролю:

1. Алгоритм визначення відстані Левенштайна та редакційного припису?
2. Особливості заповнення матриці динамічного програмування відповідно до алгоритму Вагнера-Фішера?
3. Вирішення проблеми, що пов`язана з складністю алгоритмів парного вирівнювання. Методи оптимізації алгоритмів парного вирівнювання?
4. Особливості заповнення матриці динамічного програмування для алгоритму оптимізації Міллера-Майєрса?

Комп'ютерний практикум № 12 Множинне вирівнювання біологічних послідовностей

Мета роботи: за допомогою програми ClustalOmega отримати множинне вирівнювання гомологічних послідовностей та провести аналіз вирівнювання за допомогою редактора вирівнювань JalView.

Основні задачі:

1. Ознайомитись з алгоритмами множинного вирівнювання методом динамічного програмування.
2. Ознайомитись з евристичними алгоритмами множинного вирівнювання
3. Навчитись проводити аналіз результатів множинного вирівнювання за допомогою програми JalView

Теоретичні відомості

Вирівнювання амінокислотних або нуклеотидних послідовностей - це процес зіставлення порівнюваних послідовностей для такого їх

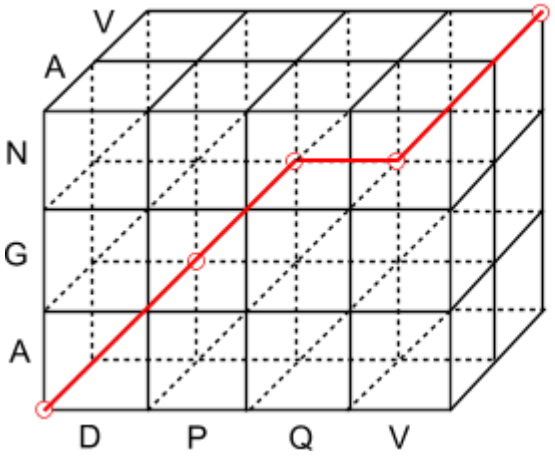
розташування, при якому спостерігається максимальна кількість збігів амінокислотних або нуклеотидних залишків. Множинним (Multiple sequence alignment (MSA)) називають вирівнювання трьох або більше біологічних послідовностей. Для побудови оптимального множинного вирівнювання послідовностей у відповідні стовпці записують якомога більше подібних знаків. Множинне вирівнювання використовується в молекулярній біології для вирішення таких завдань:

a) пошуку у послідовностях мотивів, які характеризують білкові сімейства;
b) пошук гомологій між невідомою послідовністю та відомими сімействами послідовностей;
c) передбачення вторинної або третинної структури біологічних макромолекул
d) вибору праймерів для ПЛР;
e) побудови філогенетичних дерев;

Груповий аналіз послідовностей, що входять до сімейства генів, передбачає встановлення зв'язків між більш ніж двома членами групи, що дозволяє виявити приховані консервативні характеристики сімейства. Метою множинного вирівнювання послідовностей є отримання інформації про структуру послідовностей, на підставі якої можна прийняти рішення про приналежність вирівняних послідовностей до відповідних білкових сімейств. У порівнянні з парним вирівнюванням, результат множинного вирівнювання дає більше інформації про еволюційну консервативність макромолекул. Для того щоб множинне вирівнювання було максимально інформативним, воно повинно містити рівномірну вибірку близько та віддалено схожих послідовностей -гомологів. Множинне вирівнювання групи послідовностей може забезпечити інформацію про найбільш подібні ділянки, що властиві даній групі. У білках такі області можуть бути

представлені консервативними доменами, що виконують певну функцію. Якщо відома просторова структура однієї або декількох послідовностей вирівнювання, то іноді можливо ідентифікувати амінокислоти, що утворюють подібні просторові структури в інших білках – членах вирівнювання [18].

Метод динамічного програмування використовують не тільки для парного вирівнювання. При множинному вирівнюванні n послідовностей необхідно побудувати багатовимірну матрицю. На рисунку 31 показана схема зворотного проходу для вирівнювання трьох послідовностей. Кожну комірку матриці записують вказуючи три координати (x, y, z). Шлях в матриці (на рисунку 31 позначено червоним кольором), що задається стартовим положенням з координатами $(0,0,0)$ та закінчується коміркою матриці з координатами m, n, o , відповідає оптимальному глобальному вирівнюванню трьох послідовностей. Процедуру зворотного проходу виконуємо послідовно по коміркам матриці, що мають координати: $(4,3,2) \rightarrow 3,2,1 \rightarrow 2,2,1 \rightarrow (1,1,1) \rightarrow (0,0,0)$.



Оптимальне множинне вирівнювання:

DPQV
AG - N
A - - V

Рисунок 31 – Схема процедури зворотного проходу для трьох послідовностей

Використання методу динамічного програмування для отримання оптимального множинного вирівнювання біологічних послідовностей має суттєві недоліки. Наприклад, для отримання множинного вирівнювання семи послідовностей $(N = 7)$ білків-глобінів, що мають довжину приблизно по 150 амінокислотних залишків, алгоритму необхідно пройти 10^{15} вузлів матриці [10]. Якщо послідовності, що вирівнюються, мають однакову довжину L, складність алгоритму багатовимірного динамічного програмування по пам`яті буде $O(L^N)$, а по часу - $O(2^N L^N)$. При збільшенні кількості послідовностей вже при $N \geq 10$, ефективно вирішити задачу множинного вирівнювання послідовностей буде надзвичайно складно. Враховуючи такі суттєві обмеження у застосуванні алгоритму багатовимірного динамічного програмування, було запропоновано підхід, що дозволяє досить ефективно вирівняти 5-7 послідовностей довжиною 200-300 амінокислотних залишків [19]. Цей алгоритм дозволяє знайти оптимальний шлях вирівнювання без попереднього аналізу великої кількості вузлів матриці. Програма *MSA,* що реалізує покращений варіант алгоритму множинного вирівнювання не знайшла широкого застосування для проведення оптимального множинного вирівнювання [20].

Одна з найбільш популярних програм для множинного вирівнювання – Clustal Omega (та її модифікації). На рисунку 32 приведена графічна схема методу прогресивного множинного вирівнювання:

1. На самому початку програма будує всі парні вирівнювання аналізованих послідовностей;
2. Відповідно до парних вирівнювань отримують матрицю еволюційних дистанцій;
3. Формується кластерна діаграма (філогенетичного дерево), що побудована відповідно до матриці еволюційних дистанцій;

4. Відповідно до філогенетичного дерева програма будує множинне вирівнювання з використанням алгоритму глобального вирівнювання методом динамічного програмування.

Після проведення парного (глобального) вирівнювання кожної послідовності зі всіма, на наступному етапі розраховується дистанція між вирівняними послідовностями. Найпростіший спосіб визначення генетичної відстані базується на підрахунку позицій в вирівнюванні де спостерігаються заміни. Ультраметричне філогенетичне дерево дозволяє зробити висновки про еволюційне походження даної групи послідовностей та відібрати пари, для яких буде проведено так зване довирівнювання.

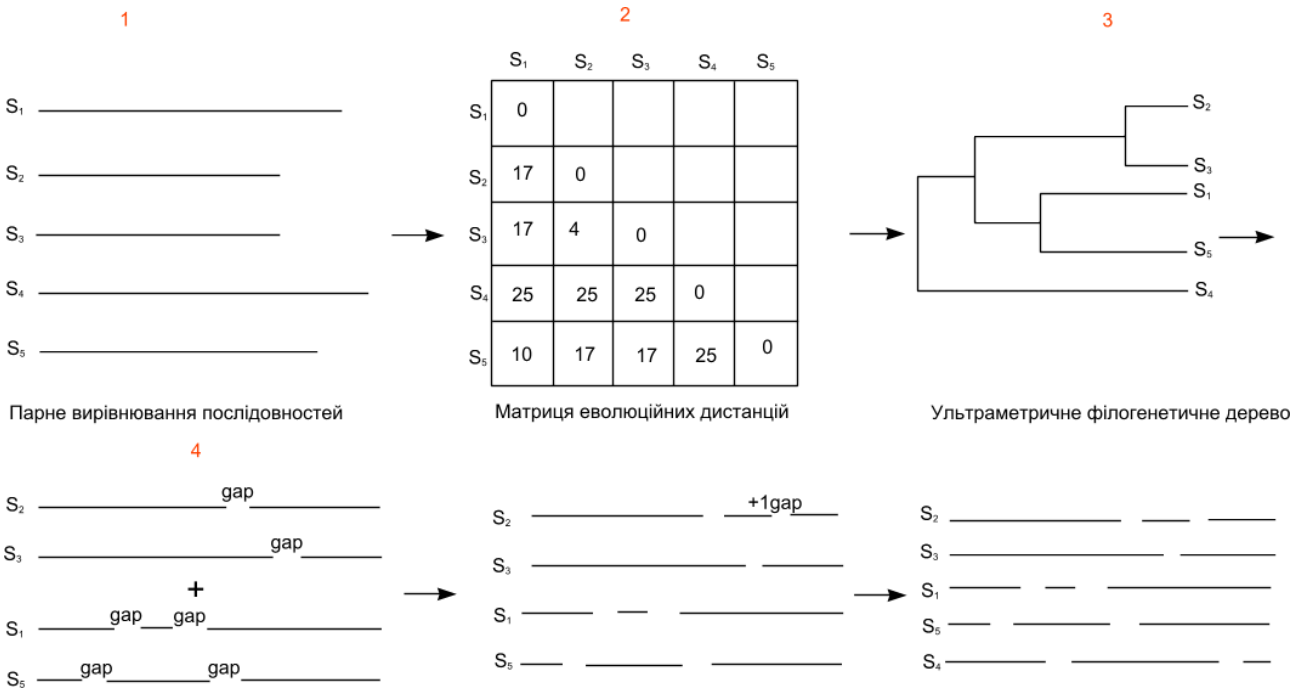


Рисунок 32 – Схема прогресивного множинного вирівнювання послідовностей

При цьому, до результатів парного вирівнювання еволюційно близьких послідовностей S_2 та S_3 записують результат вирівнювання S_1 та S_5. До вирівняної групи послідовностей S_1, S_2, S_3 та S_5 записують

послідовність S_4 (див. рисунок 32). Для отримання оптимального множинного вирівнювання послідовностей зазвичай використовують такі вагові системи оцінки збігів:

1. Послідовності записують так, щоб максимізувати кількість символів, що не відрізняються один від одного (використовують для близьких гомологів)
2. Вага стовпчиків вирівнювання оцінюється функцією суми пар (SP-Score), що враховує матриці замін амінокислот PAM та Blosum. Вирівняні послідовності записуються так, щоб максимізувати вагу вирівнювання.
3. Ентропійна система оцінки. Оптимальне вирівнювання мінімізує ентропію. Ентропію кожного символу у стовпчику вирівнювання можна визначити за допомогою рівняння Шеннона:

$$H = - \sum_{x=A,T,G,C} p_x \, log p_x$$

Значення ентропій для кожного символу вирівнювання додаються. Оптимальним буде те вирівнювання, в якому сумарне значення ентропії відповідних стовпчиків буде мінімальним. Для чотирьох фрагментів нуклеотидних послідовностей записане оптимальне вирівнювання (рисунок 33). Будь-яка трансформація даного фрагменту вирівнювання буде збільшувати ентропію стовпчиків вирівнювання.

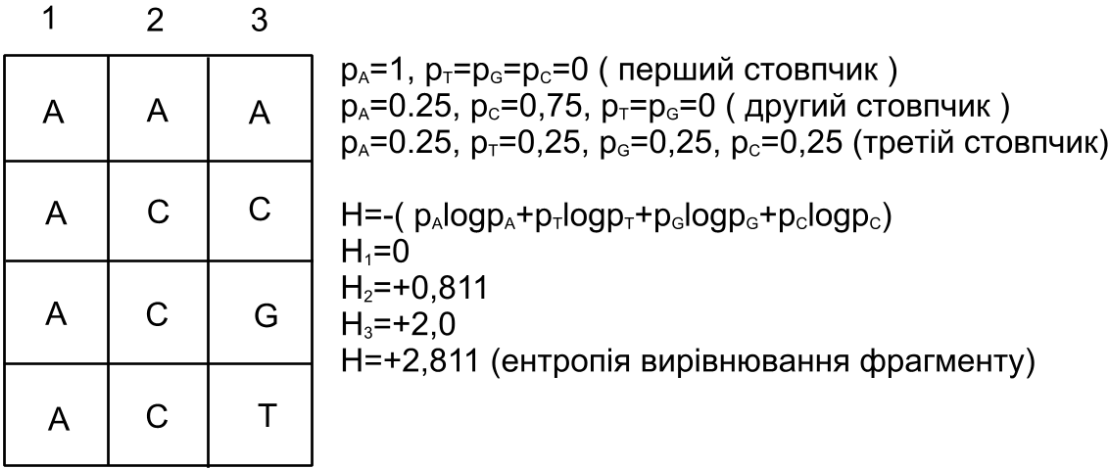


Рисунок 33 – Оптимальне множинне вирівнювання фрагменту нуклеотидних послідовностей

Для полегшення аналізу результатів множинного вирівнювання білків амінокислотні залишки різних типів забарвлюються різними кольорами. Один з можливих способів забарвлення (для програми Clustal Omega) представлений у таблиці 11. Якщо у відповідному стовпчику множинного вирівнювання відбувається заміна амінокислот, що належать одній групі, то заміна вважається консервативною. Результати вирівнювання можна завантажити за допомогою редактора вирівнювань JalView для зручного аналізу множинного вирівнювання біологічних послідовностей.

Таблиця 11 - Забарвлення амінокислотних залишків при візуалізації множинного вирівнювання білкових послідовностей програмою Clusrtal W

Амінокислоти	Тип амінокислотного. залишку	Позначення кольору
AVFPMILW	Неполярні залишки	червоний
DE	Гідрофобні	синій
RHK	Полярні	фіолетовий

STYHCNGQ	Негативно заряджені	зелений
Інші		сірий

На рисунку 34 показано результат множинного вирівнювання ендонуклеази коня (RNAS1_HORSE), кита - полосатика (RNAS1_BALAC) та великого рудого кенгуру (RNAS1_MACRU). Важливим елементом візуалізації результатів множинного вирівнювання є анотації або консенсуси (консенсусний рядок). Найпростіший вид анотації використовує програма ClustalOmega, в якій ступінь консервативності фізико-хімічних властивостей амінокислот в даній позиції вирівнювання (в даному стовпці) позначається символами: "*" - ідентичність, ":" - консервативність та "." - напівконсервативність замін амінокислот в межах даного стовпчика вирівнювання.

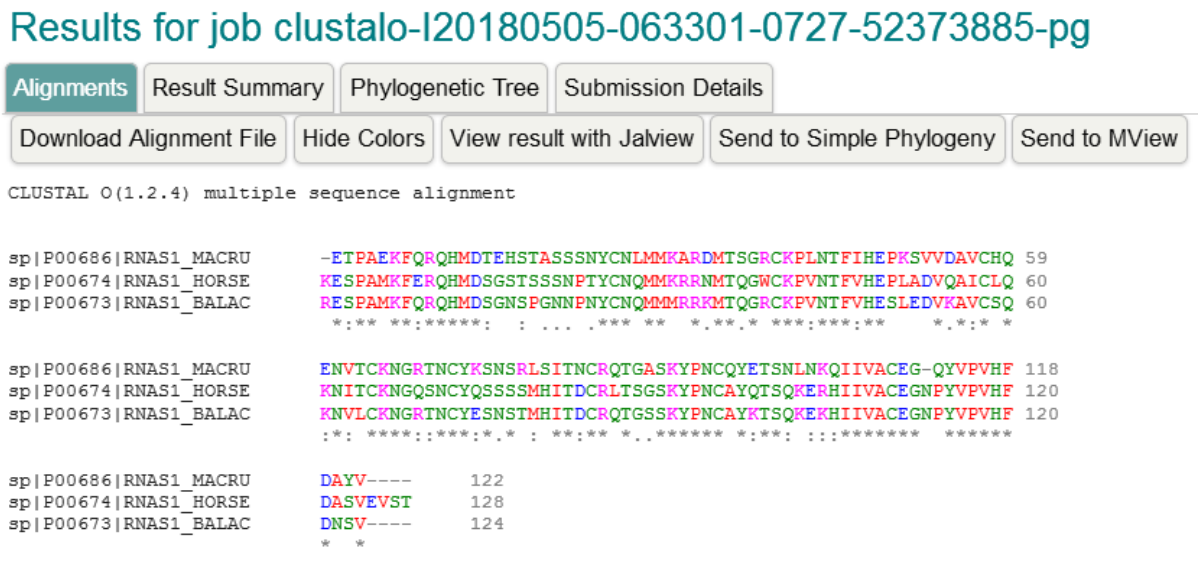


Рисунок 34 - Результат множинного вирівнювання амінокислотних послідовностей за допомогою сервісу Clustal Omega

Порядок виконання роботи

- За допомогою програми BlastP на сервері NCBI (blast.ncbi.nlm.nih.gov) знайдіть в SwissProt п'ять-десять різних гомологів заданого білка (див. файл *Student protein bacteria .xls*).Пошук необхідно проводити тільки в таксоні Bacteria. Необхідно отримати вибірку послідовностей з наступними властивостями:

2-3 послідовності, що на 60-70% збігаються з фрагментом заданої послідовності;

3-4 послідовності, що на 50-60% збігаються з фрагментом заданої послідовності;

знайти мінімум одну послідовність, що збігається із заданою приблизно на 35-45%

На сторінці видачі програми blastP після слова Alignments є декілька корисних закладок, які необхідно використовувати при виконанні першого завдання. Спочатку необхідно вибрати знахідки з відповідним відсотком збігів, поставивши галочки у віконці біля назви послідовності. Потім вибираємо закладку <Distance tree for results>, що розташована після параметру Alignments. Відкриється віконце з філогенетичним деревом обраних послідовностей. Якщо серед представлених послідовностей буде ідентифікований занадто близький гомолог, необхідно його видалити та залишити тільки одну з гомологічних послідовностей. Для цього необхідно повернітьсяся на головну сторінку видачі та зняти виділення з відповідної послідовності. Виберіть замість видаленої іншу знахідку з меншим відсотком ідентичності та перевірте, чи не є вона близьким гомологом. Після того, як вибір гомологів завершений необхідно отримати FASTA формати обраних послідовностей. Для перегляду анотованого звіту по обраним послідовностям у БД Genbank необхідно натиснути кнопку <GenPept>. На сторінці, що відкрилася в меню Display виберіть FASTA, а

в меню Send to – вибрати File, та зберегти обрані послідовності у форматі FASTA.

➢ Побудуйте множинне вирівнювання створеної вибірки гомологічних послідовностей за допомогою програми Clustal Omega (http://www.ebi.ac.uk/services). Для гомологічних білків побудувати парне вирівнювання. Результати вирівнювань зберегти у звіті. Проаналізуйте отримані вирівнювання та заповніть таблицю 12.

Таблиця 12

Кількість однакових амінокислот	
Кількість консервативних амінокислот	
Кількість напівконсервативних амінокислот	
Кількість гепів	

За допомогою програми JalView (http://www.jalview.org/download) необхідно проаналізувати результати множинного та парного вирівнювання послідовностей. Серед досліджуваної вибірки послідовностей визначити найбільш близького гомолога відповідно до заданого білка. Яке значення E-value має знайдений «близький» гомолог ?

Питання до самоконтролю:

1. Основні етапи прогресивного евристичного алгоритму множинного вирівнювання ?
2. Вагові системи оцінки збігів при множинному вирівнюванні?
3. Характеристика методів визначення еволюційних дистанцій?
4. Методи побудови ультраметричних дерев. Метод UPGMA?

5. Характеристика консенсусного рядка програми ClustalOmega?

Додаток А

Оформлення звіту за результатами виконання комп'ютерного практикуму

Звіт по виконанню комп'ютерного практикуму оформляється у вигляді звітного документу.

Вимоги до звітного документу:

- усі поля параметрів сторінки – 2 см;
- верхній та нижній колонтитули – 0 см;
- текст матеріалів набирається шрифтом Times New Roman, кегль 12, міжрядковий інтервал 1,0; абзацний відступ – 1 см; вирівнювання тексту за шириною сторінки.

Звіт по комп'ютерному практикуму (Приклад 1) формується на основі протоколу, який ведеться під час виконання поточної роботи та результатів домашньої (теоретичної та практичної) підготовки.

Звіт повинен містити наступні розділи:

1. ***Вступна частина*** (назва: ВУЗу, факультету, кафедри, дисципліни; номером поточної лабораторної роботи та тема; номер групи та П.І.Б. виконавця; П.І.Б. викладача, що перевірятиме роботу);
2. ***Основна частина:***
   - мета роботи;
   - завдання до роботи;
   - розрахунки та результати поставленого завдання у вигляді отриманих результатів досліджень;
3. ***Висновки;***
4. ***Відповіді на контрольні запитання.***

Приклад 1

Факультет Біомедичної інженерії
Національного технічного університету України «КПІ»
Кафедра біомедичної кібернетики
Дисципліна «Основи молекулярної біології та біоінформатики»

Комп'ютерний практикум № ___

Тема: _____

Виконав (ла):
студент (ка) групи ___,
(ПІБ)_____

Перевірив:
викладач
(ПІБ)_____

дата _____ підпис _____

Мета роботи: _____.

Теоретичні відомості

_____
_____

Розрахунки та результати

_____
_____

Висновки

_____

Відповіді на контрольні запитання

_____
_____

Література

1. Frances S. Turner. Assessment of insert sizes and adapter content in fastq data from NexteraXT libraries//Frontiers in Genetics. Bioinformatics and Computational Biology.- January 2014 .- V. 5.- Article 5

2. Eva C Berglund, Anna Kiialainen, and Ann-Christine Syvänen. Next-generation sequencing technologies and applications for human genetic history and forensics// Investig Genetic.- 2011; 2: 23.

3. Babraham bioinformatics [Електронний ресурс]. – Режим доступу: https://www.bioinformatics.babraham.ac.uk/projects/fastqc

4. Coursera [Електронний ресурс]. – Режим доступу: https://www.coursera.org/learn/bioinformatika/home/welcome

5. David J Edwardsand, Kathryn E Holt. Beginner's guide to comparative bacterial genome analysis using next-generation sequence data// Microb Inform Exp.- 2013.- 3: 2.

6. David J. Edwards, Kathryn E. Holt. Bacterial Comparative Genomics Tutorial. [Електронний ресурс].- Режим доступу: https://holtlab.net/2015/02/25/tools-for-bacterial-comparative-genomics/

7. Maite Muniesa, Jens A. Hammerl, Stefan Hertwig, Bernd Appel, Harald Brüssow Shiga Toxin-Producing *Escherichia coli* O104:H4: a New Challenge for Microbiology// Appl Environ Microbiol. 2012 Jun; 78(12): 4065–4073.

8. Li Z. et al. Comparision of the two major classes of assembly algorithms: overlap-layout consensus and de-bruijn-graph//Breifings in functional genomics, 2012, 11(1):25-37

9. Д. В. Ребриков, Д. О. Коростин, Е. С. Шубина, В.В. Ильинский NGS высокопроизводительное секвенирование.- 2-е изд.- М.: БИНОМ. Лаборатория знаний. -2015. – 232с.

10. P. Compeau, P Pevzner, Genome Reconstruction: A Puzzle with a Billion Pieces. 2010.
11. P. S. G. Chain, D. V. Grafham, R. S. Fulton Genome Project Standards in a New Era of Sequencing// Science 2009 -326, 236
12. Московский фізико-технічний університет [Електронний ресурс]. – Режим доступу: bio.fizteh.ru/student/files/biology/biolections/lection25.html
13. Сервер комп`ютерного класу факультету біоінфоматики та биомедичної инженерії. [Електронний ресурс]. – Режим доступу: http://kodomo.cmm.msu.su/wiki/
14. Хаубольд Б., Вие Т. Введение в вычислительную биологию: эволюционный подход. – М. – Ижевск: НИЦ «Регулярная и хаотическая динамика», Ижевский институт компьютерных исследований, 2011.- 456с.
15. Maloy, S., V. Stewart, and R. Taylor. 1996. Genetic analysis of pathogenic bacteria. - Cold Spring Harbor Laboratory Press. - NY. 603 p.
16. Дурбин Р., Єдди Ш., Крог А., Митчисон Г. Анализ биологических последовательностей. – М. – НИЦ «Регулярная и хаотическая динамика», Институт компьютерных исследование. – 2006 . -480с.
17. Хабрахабр [Електронний ресурс]. – Режим доступу: https://habrahabr.ru/post/117063/
18. Огурцов А.Н. Основы биоинформатики. – Х.: НТУ «ХПИ», 2013.- 400с
19. H.Carrillo and D.Lipman. The multiple sequence alignment problem in biology. SIAM Journal of Applied Mathematics, 48:1073-1082, 1988
20. D.J.Lipman, S.F.Altschul, J.D.Kacecioglu. A tool for multiple sequence alignment. Proceedings of National Academy of Sciences, USA, 86:4412-4415,1989